File Management & Organization

Wejdan Al-Haj Abd-Alhag Adam

Khartoum-Sudan

بسم ا   الر   ن الرحيم

إدارة وتنظيم ا لفات

File Management & Organization

تأليف وإعداد وتقد   :

وجدان ا   اج عبد ا   ق آدم

Wejdan Al-Haj

فهرست الكتاب

المقدمة:

الحاجة إلى حفظ البيانات وتخزينها واسترجاعها في زمن سريع ، يتطلب لنظم تمتاز بالبرمجة الصحية والفعّالة والقوية ، ولكي نفهم ذلك لابد لنا من معرفة كيفية بناء البيانات وتخزينها والوصول إليها ومعالجتها بصورة فعالة وبالكفاءة المطلوبة.

تهتم مادة إدارة الملفات بدراسة طرق تنظيم وتخزين البيانات وكذلك دراسة وسائط التخزين المختلفة وخصائص ومميزات كل نوع وطريقة استخدامه.

وبم أن البيانات تحفظ في صورة ملفات فإن هذه المادة تهتم أيضاً بتنظيم هذه الملفات داخلياً حيث تعني بتركيبها وبنيتها وطرق الوصول إلي محتوياتها وطرق البحث والفهرسة مما يؤدي لخلق أنواع عديدة من الملفات.

أما تنظيمها خارجياً فيعني وضع كل نوع في وسط التخزين المناسب له.ولأن سرعة الوصول إلى البيانات الموجودة بالقرص أو الذاكرة الثانوية تعتبر أكثر بطئاً من سرعة الوصول إلى البيانات في الذاكرة الرئيسية نجد أن ماد إدارة وتنظيم الملفات تهتم أيضاً بكيفية تحسين سرعة الوصول إلى البيانات.

ويمكننا أن نجمل أهداف إدارة وتنظيم الملفات في الآتي:

أ.سرعة الوصول إلى المعلومات داخل الملف.
ب.الإستخدام الأمثل لوسائط التخزين الخارجي.
ج.سهولة عملية التخزين للبيانات.

مقدمة وسائط التخزين الفيزيائية:

نحتاج إلى مكان آمن لحفظ الملفات الهامة من تلف أو ضياع ، سواء كانت خاصة أو عمومية وأفضل مكان لحفظ وتخزين البيانات هو الحاسوب ، وتختلف حجم المكان المراد الحفظ فيه تبعاً لحجم البيانات المراد حفظها .

يتم تخزين البيانات في وسيط تخزين حتى تتمكن البرامج من التعامل مع هذه البيانات بالإسترجاع والتعديل وعموما يرتكز تصنيف وسائط التخزين على ثلاث عوامل أساسية:

1. سرعة الوصول للبيانات الموجودة بها.
2. التكلفة.
3. الإعتمادية أو العمر الإفتراضي.

وسائط التخزين هذه يمكن تقسيمها لنوعين أساسين هما:

أ. **وسائط التخزين الأولية**(Primary Storage) : وهي تشمل وسائط التخزين التي تتعامل مباشرة مع المعالج مثل الذاكرة الرئيسية main memory ، والcache memory وهذه الوسائط تمتاز بسرعتها العالية ولكن حجمها صغير مقارنة بالذاكرة الثانوية .

ب. **وسائط التخزين الثانوية** (Secondary Storage): وهذا النوع يضم الأقراص الممغنطة magnetic disks ، والضوئية optical disks والأشرطة الممغنطة magnetic tapes وتمتاز بسعاتها التخزينية الكبيرة وتكلفتها القليلة نسبياً ، ولكن سرعتها بطيئة مقارنة بالوسائط الأولية .

- ❖ **ملاحظة :**

البيانات المخزنة بوسائط التخزين الثانوية لايمكن للمعالج أن يتعامل معها مباشرة ولكن يجب أولاً أن تنتقل البيانات إلى وسيط التخزين الأولي .

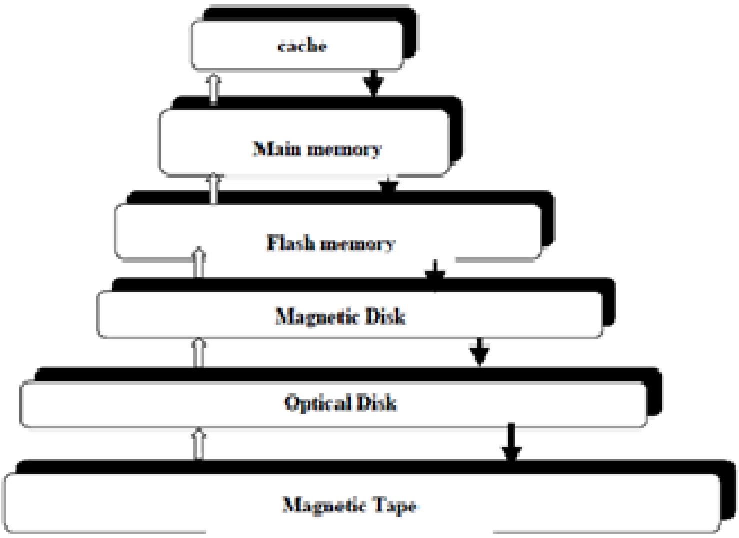
أنواع الذاكرة ووسائط التخزين (Type of Storage Media) :-

يمكن تنظيم وسائط التخزين المختلفة في شكل هرمي بناءً على سرعتها وتكلفتها وكلما انحدرنا من قمة الهرم إلى أسفل تقل السرعة والتكلفة .

ال primary storage أسرع أنواع وسائط التخزين مثل الcache والmain memory وتوجد في قمة الهرم .

وفي المستوى الثاني من الهرم توجد الsecondary storage مثل ال magnetic disks.

وفي أدنى مستوى من الهرم تأتي الtertiary storage مثل magnetic type والoptical disk & jukeboxes .



شكل(1)

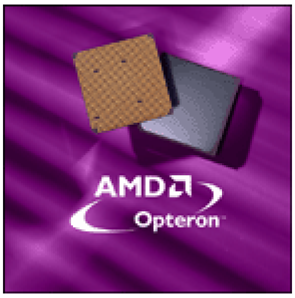
وبالإضافة لسرعة وتكلفة وسائط التخزين يوجد عامل إضافي هام وهو هل الذاكرة متطايرة أم غير متطايرة volatile or non volatile storage والذاكرة المتطايرة تعني أنها ذاكرة مؤقتة تفقد محتوياتها بمجرد انقطاع التيار الكهربي عنها ولحفظ البيانات يجب أن تكون الذاكرة غير متطايرة في الشكل الهرمي السابق (من الذاكرة الرئيسية وإلى أعلى عبارة عن volatile storage .

والأنواع أسفل ال main memory جميعها non volatile storage .

فيما يلي أنواع وسائط التخزين والذواكر حسب القسم الذي تنتمي إليه :

أ. وسائط التخزين الأوليه:-

1. الكاش ميموري Cash Memory :-



هي ذاكرة خاصة ذات سرعة عالية ، تم ابتكاراها بغرض توفير الوقت المهدور الذي يستهلكه المعالج للحصول على معلومة ما من الذاكرة الرئيسية RAM , وهي أسرع وأغلى وسائط التخزين الأولية كما إنها ذاكرة ساكنة static Ram أي ليس بها إشارة تزامن فلا يتغير محتواها ما لم أخزن بيانات أخرى عليها وعادة تدار بواسطة نظام التشغيل نفسه .

صُممت لتزود المعالج بالأوامر والمعلومات الأكثر طلبا من قبل المستخدم وهي سرّ قوة وسرعة الجهاز لتنفيذ العمليات ، أي عندما يقوم المعالج بطلب معلومة من الذاكرة الأساسية فهنالك احتمال كبير أن هذه المعلومة قد تطلب من قبل المعالج مرة أخرى ولتوفير الوقت يتم تخزينها مؤقتا ضمن ذاكرة خاصة تتميز بسرعتها العالية وتكون داخل المعالج وهي ال Cash Memoryأي أن مهمتها تخزين المعلومات الصغيرة التي قد يحتاجها البروسيسور بصفة مستمرة أثناء تشغيل الجهاز و ذلك يؤدي الى زيادة سرعة الجهاز عامة.

المعلومات والأوامر الموضوعة في ال Cash Memory يمكن الوصول إليها أسرع بعدة مرات من المعلومات الموضوعة في الذاكرة الأساسية ، فكلما استطاع المعالج الوصول إلى الأوامر والمعلومات من ال Cash Memory بشكل أسرع كلما كان الكمبيوتر يستطيع العمل بسرعة عالية أكثر .

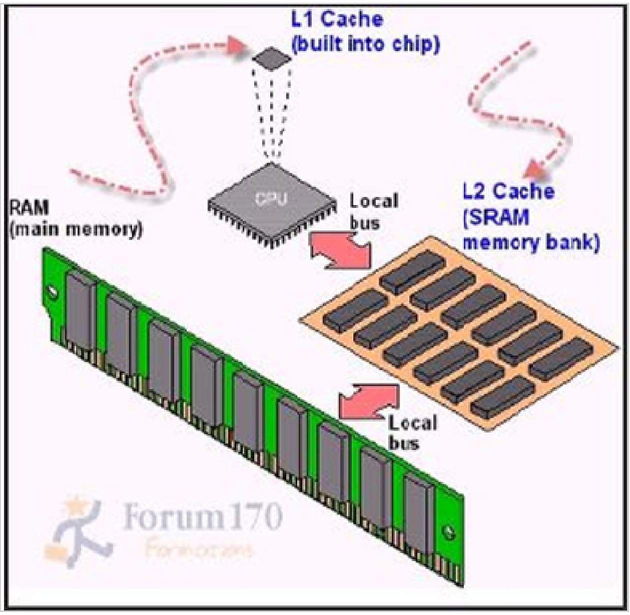
مستويات الcash memory:-

هناك ثلاث مستويات للCash Memory :

- **المستوى الأول:** ويسمى level1 وهي كاش داخلية توضع داخل المعالج الموضوعة داخل وهي أسرع المستويات وأصغرها، ومع هذا هي غالية جداً وذلك لكونها سريعة جداً حيث انها تعطي البيانات للمعالج في نفس اللحظة التي يطلبها تقريباً بدون تأخير.

- **المستوى الثاني**: ويسمىLevel2 وتوضع داخل المعالج او على المذربورد (mother bord)وسرعتها اقل من السابقة وحجمها أكبر قليلاً من السابقة .
- **المستوى الثالث**: ويسمى Level3 وهي الكاش الخارجية حيث أنها توضع غالباً على المذربورد (mother bord) وإن حجم الكاش يقاس بالكيلو بايت KB .

والشكل التالي يظهر فيه المستوى الأول والثاني للكاش(Cach) :



شكل(2)

كيف تعمل الكاش ميموري:-

عندما يجلب آمر الكاش معلومة من الذاكرة الأساسية فإنه يقوم بجلب بضع من المعلومات التي تليها ويأخذهم إلى ال Cash Memory معها ، وهذا يزيد من احتمالات تلبية المعالج بالمعلومات المطلوبة بشكل أسرع في حال طلبه المعلومة التي تلي المعلومة الأولى والتي أصبحت في ال Cash Memory مسبقاً .

قبل الوصول لذاكره النظام الأساسية RAM الذاكرتان L2 & L3 ليست مهتهم الأساسية تخزين_المعلومات فقـط و لكن أيضاً تمنع الاختنــاق الذي قد يحدث داخل المعالج بسبب تبادل المعلومات الغير هامه بين انويه المعالج .

تسجيل المعلومات على الكاش :-

يتم تسجيل المعلومات بشكل لا يقبل التكرار إطــلاقاً , بمعنى انه لا يمكن تكرار نفس المعلومة في أكثر من مستوى من ذاكرة الكاش .

سرعة الكاش ميموري :

سرعة الكاش ميموري أو التردد الذي تعمل عليه يعتمد ذلك على الموقع كما يلي:

Wejdan
Al-Haj

عندما تكون الكاش ميموري على ناقل النظام يكون ترددها نفس سرعة الناقل 66 أو 100 ميغا هرتز.

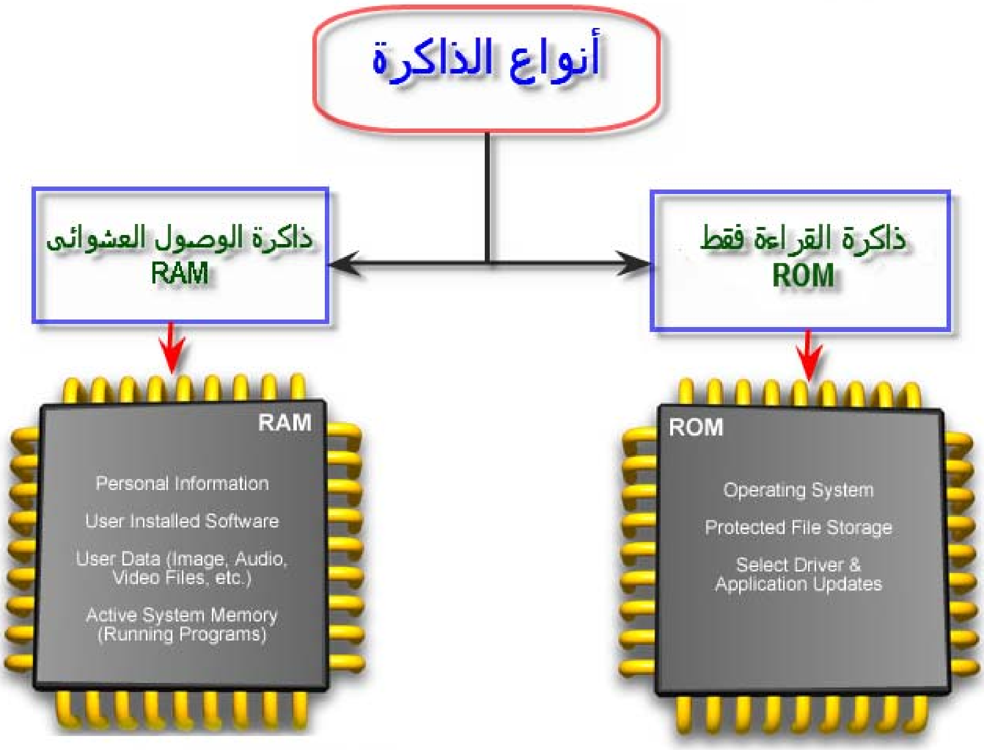
الكاش الموجودة داخل المعالج تعمل بنصف سرعة المعالج واحياناً بنفس سرعة المعالج
بعض المعالجات لها كاش ميموري من المستوى الثاني على المذر بورد وترددها لا يزيد عن 66 ميغا هرتز.

حجم الكاش :-

قطعا كلما زادت حجم الكاش كلما أمكن اختزان مزيد من المعلومات ، ولكن يؤدي هذا إلى بطء في زمن الإستجابة .

وتعتبر الCach Memory من أكثر وسائط التخزين الأولية سرعة وأقل حجما وكذلك تكلفتها عاليه جدا بالنسبة لبقية وسائط التخزين.

2. الذاكرة الرئيسية main memory:-



شكل(3)

تقوم هذه الوحدة بتخزين البيانات وتعليمات البرامج حتى تتم معالجتها ، كما تخزن نتائج المعالجة داخل هذه الذاكرة تمهيدا لنقلها الى وحدات الإخراج أي هي تمثل منطقة العمل الرئيسية للمعالج التي يحتفظ فيها بالبيانات والبرامج العاملة حاليا.

ميزتها الأساسية هي أن سعرها قليل نسبيا مقارنة مع الـcache حيث نلاحظ أن سعرها متناقص وحجمها في زيادة مضطردة ، أما عيبها الأساسي فهو أنها متطايرة volatile memory (أي أنها ذاكرة مؤقتة تفقد محتوياتها بمجرد انقطاع التيار الكهربي) ،وأيضا أبطأها نسبيا مقارنة بالـcache.

تنقسم الذاكرة الرئيسية إلى قسمين هما:-

القسم الأول : ذاكرة الوصول العشوائيRAM:-



شكل(4)

من وجهة نظر علوم الكمبيوتر يستخدم مصطلح RAM ليشير إلى نوع الذاكرة، الذي نتمكن من الكتابة فيه والقراءة منه من قِبل المعالج microprocessor وأجزاء الHardware الأخرى.

ذاكرة الوصول العشوائي هي ذاكرة الكمبيوتر الأساسية وتتألف من سلسلة من الخلايا التي تستخدم لتخزين المعلومات .

هذه الخلايا تنظم في مجموعات تسمى مواقع الذاكرة (memory locations) كل خلية لها عنوانها الخاص,والعنوان هو عبارة عن سلسلة أرقام في النظام الثنائي (1 أو 0) أو بيتات.

الكمبيوتر يمكنه أن يعنون كمية محددة من البيانات في ذاكرته الأساسية في كل مرة هذه الكمية المحدودة تعتمد على عدد البيتات التي يستطيع معالج الحاسوب معالجتها . مثلا 16_bit Processor يمكنه حمل أعظمياً 16 بت2 (بايت)من البيانات في وقت واحد.البيانات التي يعالجها الحاسب تأتيه من ذاكرة الوصول العشوائي(Ram) .

يمكن للمعالج الوصول لمكان التخزين على الرام أياَ يكن مكانه وذلك بواسطة ممر العناوين Address busses .

عندما ينجز المعالج عملية حسابية (Arithmetic operation) كالجمع والطرح فإن الأرقا |التي استخدمها المعالج يمكن إيجادها في الذاكرة.
تعمل ذاكرة الوصول العشوائي كوسيط بين المعالج ومحرك الأقراص الصلبة أو الأقراص المرنة حيث أن هذه الأقراص لا تملك السرعة الكافية لمجاراة سرعة المعالج ، لذلك يتم تخزين البيانات في وسط تخزين سريع(الذاكرة)ريثما ينتهي المعالج من معالجة البيانات وتخزينها على الأقراص الصلبة.

- جاءت تسمية هذه الذاكرة لأنها تستطيع الوصول إلى أي خلية في ا لذاكرة بمجرد معرفة الصف والعمود الموجودة فيه.
- إن رقاقة الذاكرة هي عبارة عن دائرة متكاملة تتألف من ملايين النرنستورات والرقاقات(حيث أن الترنستور والمكثف يشكلان خلية الذاكرة والتي تشكل (بت bit) ،الترانستور يعمل مفتاح تحكم فهو إما أن يقرأ حالة المكثف أو يقوم بتغييره ، أما المكثف يعمل حافظة للإلكترونات ، فعند شحن المكثف يتم حفظ قيمة واحد وعند إفراغ المكثف يتم حفظ قيمة صفر.
- إن البيانات المخزنة على هذه الذاكرة تمحى بمجرد فصل الطاقة الكهربائية عنها.

ما تأثير حجم ونوعية الذاكرة العشوائية على الحاسب بشكل عام ؟

- الأداء : يصبح الحاسب أسرع بشكل عام عند إضافة المزيد من الذاكرة ، خاصة عند التعامل مع كميات كبيرة من البيانات أو البرامج الكبيرة (البرامج الجديدة تكون أكثر تطلباً للذاكرة من البرامج القديمة)، وهذه النقطة مهمة جداً حيث أنه حتى المعالج السريع قد لا يستفاد من أقصى سرعته إذا كانت كمية الذاكرة العشوائية أقل مما يجب .
- نوعية الذاكرة العشوائية تلعب دوراً في سرعة الذاكرة وفي خيارات الترقية فيما بعد .
- قد لا يمكنك تشغيل بعض البرامج إذا كان لديك كمية قليلة من الذاكرة العشوائية .
- المشاكل والأخطاء : إن نوعية الذاكرة العشوائية تلعب دوراً في كمية المشاكل والأخطاء التي قد توجهها أثناء عملك على الحاسب ، إن قطعة ذاكرة معطوبة قد تتسبب بتوقف الحاسب المتكرر عن العمل بدون سبب واضح من الوهلة الأولى لا بل قد تذهب بعيداً وتفعل أشياء مثل تشخيص أخطاء وهمية في القرص الصلب .

ويقابل الرام ذاكرة أخري وتسمي SAM(serial access memory) هذا النوع من الذاكرة يخزن البيانات علي شكل سلسلة من خلايا الذاكرة المتتابعة مثل شريط الكاسيت فأنت لا تستطيع الوصول إلي معلومةما مخزنة في أخر الشريط إلا بالمرور علي البيانات من أول الشريط حتى تصل إلي المعلومة المطلوبة وهذا النوع بطئ جدا بالمقارنة مع الذاكرة RAM.

القسم الثاني : ذاكرة القراءة فقط ROM :-

ما هو (ROM) Read-Only Memory) ؟ هذا نوع من الذاكرة قابل للقراءة و لا تستطيع الكتابة عليها ، و البيانات المخزنة عليها يتم تخزينها في مرحلة صنع و تكوين رقاقة الذاكرة ، و هي لا توجد في أجهزة الحاسوب وحدها بل تجدها أيضا في أغلب الأجهزة الإلكترونية .

كيف تعمل الذاكرة ROM ؟

كما في الذاكرة RAM فإن الذاكرة الروم تتكون من شبكة من الصفوف و العواميد ، و لكن عند التقاء الصفوف بالعواميد نجد أن الروم مختلفة كليا عن الرام ، فحيث نجد ترانزيستور عند نقطة التقاء الصف و العمود في الرام ، نجد بدلا منه ديود diode في الروم و الذي يقوم بوصل الصف مع العمود إذا كان محتوى الخلية المتقاطعان عندها يساوي 1 ، أما إن كان المحتوي صفر فبكل بساطة لا يوجد ديود و لا يتصل الصف بالعمود عند خلية التقاطع ، و بالتالي نرى أن تشكيل رقاقة الذاكرة و تخزين البيانات عليها يتم خلال فترة التصنيع و يصبح تغيير محتوى الرقاقة مستحيل بعد إتمام التصنيع .

لماذا نحتاج أن نستعمل الروم بدلاً من الرام أو أقراص التخزين مثلاً ؟

هناك عدة أسباب لذلك :-

- البيانات المخزنة في الروم دائمة وليست معرضة للتلف بأي شكل بعكس الأشكال الأخرى من التخزين .
- البيانات المخزنة في الروم لا يمكن تغييرها بالصدفة أو عن طريق فيروس (مثلاً لا يمكن لفيروس محو المعلومات الموجودة على قرص CD-ROM) .
- المعلومات المخزنة في الروم تتوفر لأجهزة الحاسب في جميع الأوقات (رقاقة البيوس مثال جيد) .

ما هو الفرق بين RAM و ROM ؟

إن الفرق كبير وشاسع ، الذاكرة ROM (ذاكرة القراءة فقط) كما قلنا هي عبارة عن ذاكرة تخزن فيها البيانات في مصنعها و لا يمكن لمستخدم الحاسب أن يغيره بعد ذلك بل يكتفي بقراءة محتويات هذه الذاكرة ، لذا فهي تسمى ذاكرة القراءة فقط (Read Only Memory) بينما الرام تسمى ذاكرة القراءة والكتابة (أو ذاكرة الوصول العشوائية) .

في الجدول التالي نوضح أهم الفوارق بين نوعي الذاكرة :

من حيث:	RAM	ROM
يمكن الكتابة عليها بواسطة المستخدم	نعم	لا
يمكن القراءة منها بواسطة المستخدم	نعم	نعم
السرعة	أسرع	أبطأ
الاستعمالات الشائعة	مخزن مؤقت (وسريع) للبيانات التي يتعامل معها المعالج أو يتوقع أن يتعامل معها قريباً	تخزين برنامج البيوس للوحة الأم
تعرض البيانات للتلف	تمحى البيانات بمجرد إطفاء الحاسب	تبقى البيانات في الرقاقة لفترة طويلة جداً (لا نهائية تقريباً) ولا يمكن تغييرها في أغلب الأحيان

ب. وسائط التخزين الثانويه:-



تصنيف وحدات التخزين الثانوية:-

يمكن تصنيف وحدات التخزين الثانوية، بحسب الوصول إلى البيانات المطلوبة، إلى وحدات تتابعية وأخرى عشوائية.

i. وسائط التخزين الثانوية ذات الوصول المتتابع:-

يتم في هذه الوسائط استرجاع البيانات والمعلومات المخزنة بنفس الترتيب الذي سبق التخزين به .وللحصول على البيانات المطلوبة في هذا النوع من الوسائط لابد من المرور على كل ما سبقها من البيانات، وبنفس الترتيب إلى أن يصل إلى البيانات المطلوبه، الأمر الذي يجعله يستغرق وقتا أطول للوصول إلى البيانات المطلوبة ،و مثال لوسائط التخزين الثانوية ذات الوصول المتتابع هو الشريط المغناطيسيMagnetic Tape .

1- الأشرطة الممغنطة Magnetic Tape:-

يعتبر الشريط المغناطيسي Magnetic Tape من أهم وأقدم وسائط التخزين الثانوية ذات الوصول المتتابع، والذي يقوم بحفظ البيانات على شريط مغناطيسي قابل للمغنطة .و يحتوي هذا الشريط على بقع مغناطيسية تُعبر عن المعلومات التي ستخزن عليه، حيث يمثل البقع الممغنطة قيمة 1 ، والبقع غير الممغنطة قيمة 0 و يعتبر أهم استعمال للشريط المغناطيسي احتفاظه بنسخ احتياطية للبيانات والبرمجيات التي نخشى عليها من الضياع، وذلك بالقيام بإجراء نسخة احتياطية لهذه البيانات على سطح الشريط.

استخدامات الأشرطة المغناطيسية:-

تستخدم الأشرطة المغناطيسية كوسيلة تخزين إضافي لحفظ قدر كبير من كميات المعلومات، وذلك بسعر رخيص جدًا مقارنة بالذاكرة الرئيسة .غير أن وقت الحصول على البيانات، والذي يعرف باسم Access-Time، يعتبر كبيرا بالمقارنة مع الوسائط الأخرى .ولذلك فإن الأشرطة المغناطيسية لا تفضل في الاستعمال بمصاحبة نظم الحاسوب فائقة السرعة في الأداء.

والأشرطة المغناطيسية يمكن استخدامها لأكثر من مرة لتسجيل البيانات .وذلك بمحو البيانات السابقة وتسجيل البيانات الجديدة على نفس السطح .وأهم ما تتميز به الأشرطة المغناطيسية هو أن البيانات المسجلة عليها لا تختفي بطول الزمن.

أنواع الأشرطة المغناطيسية:-

١. شرائط البكرات
٢. البكرات الصغيرة
٣. البكرات المصغرة
٤. الأشرطة المعلبة
٥. أشرطة QIC و أشرطة DAT

جهاز تشغيل الأشرطة المغناطيسية:-

يستخدم جهاز تشغيل الأشرطة المغناطيسية في قراءة وكتابة البيانات على الأشرطة المغناطيسية، بواسطة رؤوس القراءة والكتابة ويتميز هذا الجهاز بسرعة القراءة و الكتابة، حيث تتراوح سرعته من 15 إلى 1250 رمزًا في الثانية، وذلك حسب سرعة حركة الشريط وكثافة التسجيل .وتتكون منظومة الشريط المغناطيسي من خمسة أجزاء رئيسة هي:

١. شريط مغناطيسي.
٢. محرك كهربي يعمل على تحريك الشريط المغناطيسي، وذلك بإدارة البكرات ميكانيكيًا بحيث يمر الشريط المغناطيسي أسفل رأس التسجيل المغناطيسي .وتتم الحركة بناء على الأوامر الصادرة من وحدة التحكم بالحاسوب.

٣. نظام التسجيل ويسمح بالكتابة والقراءة على الأشرطة المغناطيسية .ويتكون هذا النظام من مكبر إشارة إلكتروني يعمل على تكبير نبضات الكتابة والقراءة، ومترجم يقوم بتحويل إشارات الشريط إلى نبضات أرقام ثنائية يفهمها الحاسوب بوحداته المختلفة.

٤. دارة ربط ونقل البيانات وتتكون هذه المنظومة من الأجهزة اللازمة لاختيار مكينة تشغيل الشريط المناسب، وذلك في حالة وحدات إدارة الأشرطة المغناطيسية المتعددة الأشرطة، وكذلك الدارات الإلكترونية اللازمة لاستقبال بيانات الشريط ومن ثم دفعها إلى وحدات الحاسوب للمعالجة.

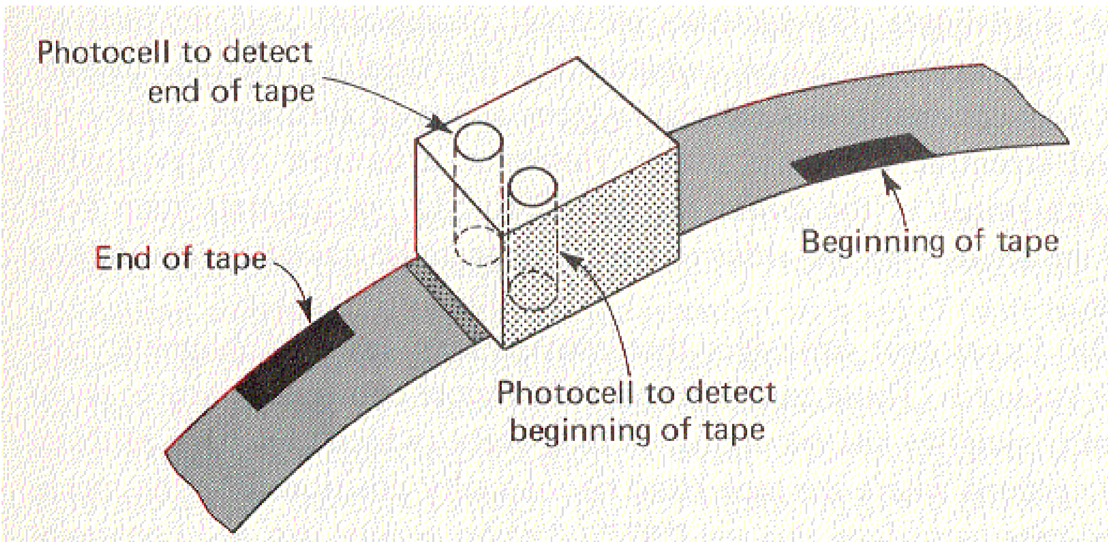
٥. نظام إستشعار يستخدم خلية ضوئية كهربية في بداية ونهاية كل شريط و يوجد جزء منه غير مطلي بطبقة المادة القابلة للتمغنط، وذلك لتأدية مهمتين :

المهمة الأولى : هي السماح بلف جزء من أول ونهاية الشريط حول قرص دوران البكرة.

المهمة الثانية : هي استخدام الجزء الباقي من الشريط بدون طبقة مغناطيسية لطلائه بطبقة معدنية غير حديدية وعاكسة للضوء، وذلك لإظهار بداية ونهاية الجزء الصالح من الشريط للتسجيل والقراءة.

ولكشف ملصق عواكس بداية ونهاية جزء الشريط للتسجيل والقراءة تستخدم خليتان ضوئيتان كهربائيتان لاستشعار وتحديد موقع هذين الملصقين.

إن استشعار ملصق بداية الجزء الممغنط من الشريط يصدر إشارة إلى وحدة تشغيل البكرات ببداية عملية التسجيل من كتابة وقراءة على حين أن استشعار ملصق الجزء الممغنط من الشريط يصدر إشارة بأمر إلى وحدة تشغيل البكرات لإيقاف دوران البكرة.



شكل(5)

فعندما تقوم البرامج بإجراء النسخ الاحتياطية تقوم بإرسال تيار من المعلومات إلى رأس الكتابة Write Head الذي يتركب من ثلاثة أجزاء حيث يكون Read-while-Write Head هو الذي يتوسط رأسي القراءة، فإن الشريط المغناطيسي يقوم بالتأكد من

المعلومات التي تم تسجيلها، وذلك بأن يقوم بقراءتها بواسطة أحد الرؤوس ويتم اختيار الرأس حسب اتجاه الشريط، فيقوم بمطابقة هذه المعلومات بالتي تم إرسالها إلى رأس الكتابة، وإن كان هناك خطأ فإنه يقوم بإعادة كتابتها على جزء آخر من الشريط المغناطيسي.

وتوجد قرب نهايات الشريط ثقوب تقوم بإعلام جهاز التشغيل بأن يتوقف لان الشريط قد وصل إلى نهايته، وذلك عوضًا عن استعمال اللاصقة المعدنية .

حلقات حماية شريط البكرات:-

لأن التسجيل على الشريط الممغنط يؤدى إلى مسح ما هو مسجل عليه، لذلك يجب التمييز بين الشريط الممغنط المسموح التسجيل عليه من غيره، وذلك بوضع حلقة حماية محتويات الشريط على السطح السفلي لبكرة الشريط .أما الأشرطة الأخرى التي تحتوى على بيانات أو معلومات أو برامج مطلوب الاحتفاظ بها، فيتم وضع هذه الحلقة في موقعها لتوفير الحماية لما هو مسجل عليها بحيث لا يسمح بعملية التسجيل .ويكون تسجيل البيانات على الشريط المغناطيسي على شكل بقع مغناطيسية، حسب شفرة خاصة، بحيث تستطيع رؤوس القراءة والكتابة الإحساس بهذه البقع أثناء القراءة، وتقوم بإرسال نبضات متناسبة مع المعلومات على سطح الشريط.

تنظيم الملفات على الأشرطة المغناطيسية:-

تسجل البيانات على الأشرطة المغناطيسية على هيئة ملفات بأحجام مختلفة .وتُقسم الأشرطة والملفات إلى أنواع ثلاثة هي:-

- **شريط متعدد الملفات:** وذلك في حالة ملفات حجم البيانات فيه قصيرة حيث يتم تسجيل أكثر من ملف واحد علي نفس الشريط. ويوجد مميزات خاصة تستعمل كشفرة للدلالة علي بداية ونهاية كل ملف منفصل.

- **شريط الملف المفرد:** وذلك في حالة البيانات المتوسطة الحجم والتي تشغل بكرة واحدة كملف لها.

- **ملف الأشرطة المتعددة:** وذلك في حالة البيانات الكبيرة الحجم مما يتطلب أكثر من بكرة مغناطيسية لتسجيل البيانات .وللتفريق بين البكرات المتعددة تستخدم ملصقات لتحديد موقع كل منها من ملف البيانات، وذلك علاوة علي ملصق بداية ونهاية البكرة.

مميزات وعيوب الشريط المغناطيسي:-

١. الشريط المغناطيسي يناسب التطبيقات ذات الملفات التتابعية التي تتطلب سرعة في عمليات القراءة والكتابة وطاقة تخزين عالية.
٢. طول السجل غير ثابت.
٣. يمكن استخدام الشريط المغناطيسي كوسيط إدخال أو إخراج أو كليهما.

٤. يمكن قراءة المعلومات المسجلة على الشريط أكثر من مرة.
٥. تستمر البيانات على الشريط لفترة زمنية إلا إذا تمت إزالتها بواسطة عملية المسح.
٦. الكتابة ثانية على الشريط تمسح المعلومات السابقة المسجلة عليه.
٧. تستخدم الأشرطة المغناطيسية غالبًا عندما تكون المعلومات متتالية بطبيعتها.

ii - وسائط التخزين الثانويه ذات الوصول العشوائي:-

في هذا الصنف من وسائط التخزين يتم الانتقال إلى البيانات المطلوبة منها بالتوجه مباشرة إلى موضع التخزين، دون المرور على ما يسبقه في مواضع تخزينية ومن أمثلة ذلك : الأقراص المغناطيسية Magnetic Disks والأقراص الليزريه , Laser Disks،وتُعتبر الأقراص الممغنطة أو المغناطيسية أكثر وسائط التخزين شيوعا واستخدامًا ،وتتميز بقدرتها الاستيعابية العالية، وسرعة تداول المعلومات المخزنة عليها.

1-الفلاش ميموري flash memory:-

تعرف أيضا ب(EEPROM) وهي اختصار:

(Electrieally Erasable Programmable Red Only Memory)



شكل(6)

وهي نوع وسط بين القرص والذاكرة الرئيسية ، وتختلف عن الذاكرة الرئيسية في أنها غير متطايرة وهي الأسرع في استرجاع البيانات من الذاكرة الرئيسية .

ويعتبر الفلاش ميموري هي الأكثر استخداما بدلاً عن القرص في أنظمة الكمبيوتر الصغيرة المضمنة في العديد من الأجهزة والتي تتطلب وسائط تخزين كبيره جدا.

ذاكره الفلاش تختلف عن الذاكرة الالكترونية العادية في الحاسب الشخصي في أنها تستطيع الاحتفاظ بالبيانات المسجلة عليها حتى في حاله انقطاع التيار الكهربائي عنها أو عند نزع البطارية في الجهاز الالكتروني المثبت بها.

- **بدون أجزاء متحركة** : لا يوجد بوحدات ذاكره الفلاش أجزاء متحركة كما هو الحال في وحدات الاسطوانات أو وحدات التخزين الرئيسية، عدم وجود أجزاء متحركة يقلل من احتمالات الأعطال بدرجه كبيره وبذلك تقل المشاكل أثناء الاستخدام.
- **سرعه الوصول للمعلومات**: نظرا لعدم وجود حركه ميكانيكيه بوحدات ذاكره الفلاش وأن التعامل مع البيانات يتم بالكامل بطريقه الإلكترونية فإن سرعة قراءة وكتابه المعلومات تكون أسرع بكثير من وحدات التخزين الأخرى.
- **سهوله تخزين الملفات** : لا يوجد بوحدات ذاكره الفلاش القيود التي توجد علي اغلب أنواع الاسطوانات المدمجة في عدد مرات الكتابة عليها فنحن نتعامل مع هذه الوحدات بنفس سهوله التعامل مع وحدات التخزين الرئيسية. كما أنها تتميز علي الاسطوانات المدمجة CD في أنها لا تحتاج الى برامج خاصة للتسجيل عليها أو للعمل معها.

أسباب تلف الفلاش :-

١- التحميل الخاطىء لملف software جديد يؤدى إلى أن تكون كل المعلومات المسجلة على الفلاش ميموري غير صحيحة وبالتالي لا يستجيب الجهاز لأي اتصال مع جهاز الكومبيوتر .
٢- قطع التيار الكهربي أثناء عملية تحميل ملف اللودر المدمج software على الجهاز.
٣- حدوث تغير في التيار الكهربي إثناء الاستخدام العادي .
٤- تراكم الأتربة على وحدة الفلاش ميموري.

١- الأقراص الضوئية (optical disk):-



شكل(7)

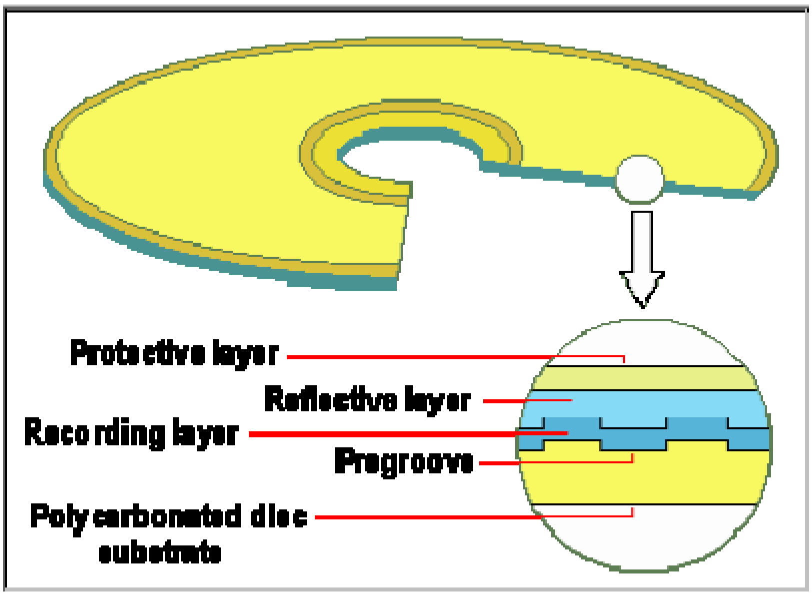
أهم مميزات الأقراص الضوئية السعه الضخمة والكلفة الرخيصة، مما تجعلان هذا النوع من الوسائط الأمثل لتخزين كميات البيانات الكبيرة نسبيًا .وتوجد أنواع عديدة من الأقراص الضوئية، وتختلف عن بعضها من حيث برمجة محتوياتها، ومنها:

١. CDROM: تعني قرص مدمج قابل للتسجيل فقط :وهذا النوع هو الاكثر استخداما في وقتنا الحالي حيث انه هو الاول المدعوم من قبل مشغلات الاقراص .يمكن تخزين البيانات على سطح القرص الضوئي مرة واحدة بواسطة أجهزة خاصة.

٢. CD RW: قرص ضوئي يمكن برمجته ومسحه عدد من المرات .وتعني قرص مدمج قابل لإعادة التسجيل :وهذا النوع يكون في العادة اغلى من قرصCD-R ويتميز بقابلية إعادة استخدامه أكثر من مرة ويمكنك حذف محتوياته وتقريبا يحدد عمر استعمال القرص في الكتابة وإعاده الكتابة ب 1000 مرة ولكن يوجد عيب في هذه الاقراص انه ليست كل المشغلات تستطيع ان تشغل هذه النوعية وخصوصا القديمة.

ويجب أن يتناسب مشغل الأقراص الضوئية مع نوعية القرص الضوئي وكذلك العملية المطلوبة، من حيث كونها قراءة أو كتابة.

مشغل الأقراص الضوئية:-



شكل(8)

إن مشغل الأقراص الضوئية CD Rom Drive مخصص للبيانات لا يحتوي على أية أزرار أو ما شابه باستثناء زر تحميل وتفريغ .ويتم التحكم في عملية القراءة من القرص بواسطة برمجيات موجودة في الحاسوب وذلك بإرسال أوامر إلى دارة جهاز التحكم . وتكون هذه البيانات مرتبة في دوائر متحدة المركز (مسارات) ومقسمه شعاعيا إلى قطاعات .ويدور القرص بنفس معدل السرعة الزاوية الثابتة، ويجب أن تكون هذه القطاعات أكبر ما يمكن لتستوعب نفس كمية المعلومات التي تستوعبها القطاعات الداخلية.

ويقوم محرك بتغيير معدل سرعة دوران القرص باستمرار، ويسلط الليزر شعاعًا مرَكزًا بواسطة ملف البؤرة .وتولد كل نبضة ضوئية تصطدم بثنائي إحساس الضوء فولتية كهربائية خفيفة لتوليد سلسلة من أرقام (0,1) التي يستطيع الحاسوب ترجمتها وفهمها . ومعدل نقل البيانات بين مشغل الأقراص والحاسوب هو أقل من 1.2 مليون ثنائية في الثانية، باستخدام المواجهات القياسية.

يعتبر CD-I مجموعة من المواصفات المتكاملة من الدارات الإلكترونية والبرمجيات، والتي تتكون من مشغل الأقراص الضوئية و معالج وكذلك دارات لمعالجة الصوت والصورة .

كيف يتم تخزين البيانات:-

يتم تخزين البيانات كسلسلة من الثنائيات على مسار حلزوني واحد يبدأ من مركز الأسطوانة ويمتد نحو حافته الخارجية. وتركز أشعة القراءة الليزرية على طبقة البيانات ضمن الأسطوانة البلاستيكيه حيث تتناوب التجاويف Pits على الأرضية Land والأرضية عبارة عن منطقة ملساء خالية من التجاويف .يرتد الضوء المنعكس من خلال منشور prism وينعكس على حساس ضوئي يتغير توتر خروجه اعتمادا على كمية الضوء التي يتلقاها .وكما هو الحال في الوسط المغناطيسي لا تمثل التجاويف الأرضية بشكل مباشر الأصفار والواحدات، بل إن الانتقالات بين التجاويف والأرضية هي التي تمثل البيانات.

وعند تسليط الضوء على تجويف فإنه يتناثر بشكل أكبر من تناثره عند تسليطه على الأرضية .ويستطيع رأس القراءة بهذه الطريقة تحسس الانتقالات بين التجاويف في المسار ويمكنه بالتالي إعادة توليد البيانات .وتخزن البيانات في عناصر صغيرة جدًا، يبلغ طول الخطوة المسارية Track Pitch أي المسافة بين المسارات المتجاورة 1.6 ميكرون فقط، الميكرون هو واحد بالألف من الملليمتر .ويتم طبع التجاويف في مساحة فارغة من البلاستيك متعدد الكربونات يتم تغطيتها بطبقة رقيقة من الألومنيوم الذي يعطي الأسطوانة لونها الفضي المميز .ثم تغطى طبقة الألمنيوم بطبقة رقيقة من الورنيش الذي يؤمن سطحا أملسًا يمكن طباعة عنوان الأسطوانة عليها.

لماذا تتلف أسطوانة الليزر؟؟؟

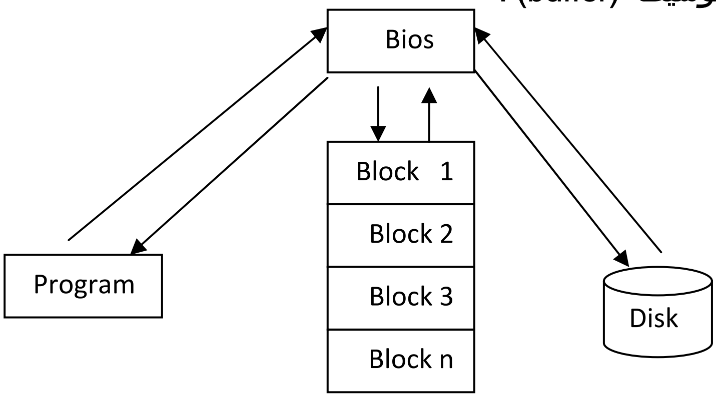
يجهل العديد من المستخدمين أن الطبقة العلوية من أسطوانات CD وهي الطبقة التي يطبع عليها عنوان ومحتويات الأسطوانة هي في الواقع أكثر عرضة للتلف من الطبقة السفلية ذات السطح الصافي .وإذا خدش السطح العلوي بعمق كاف لتلف طبقة الألومنيوم العاكسة فليس أمامك من وسيلة لإنقاذ هذا الأسطوانة سوى استبدالها .وتركز أشعة الليزر في الواقع من ناحية أخرى على طبقة تقع ضمن القاعدة الصافية للأسطوانة ويمكنها قراءة البيانات متجاوزة بعض الخدوش الصغيرة على السطح .وحتى إذا كان الخدش حادا لدرجة أنه يمنع أشعة الليزر من قراءة البيانات، فمن الممكن أن نتمكن من إنقاذ هذه الأسطوانة عن طريق تنظيفها وتلميعها .

2- الأقراص الممغنطه (Magnetie Disks):-

الأقراص المغناطيسية تستخدم لحفظ البيانات الكبيرة وتعتبر هي الوسيط الأساسي لتخزين البيانات لفترات طويلة وحجم القرص يحدد بسعته بالbyte مثلاً القرص المرن floppy disk سعته 1.4 mb ، والقرص الصلب سعته في تزايد مستمر وتتراوح الآن من بين بضعة mb إلى عشرات الG byte .

لمعالجة البيانات يتم نقلها أولاً من القرص إلى الذاكرة الرئيسية وبعد المعالجة تعاد إليه ثانية ليتم حفظها فيه ، وتمتاز الأقراص المغيناطيسية بأنها غير متطايرة وتحتفظ ببياناتها رغم انقطاع التيار الكهربي عنها أو تعطل النظام ونادراً ما تصاب بأعطال تؤدي إلى فقدان البيانات فيها .

الخازنة الوسيطة (buffer) :



شكل(9)

- BIOS : هي شريحة إلكترونية صغيرة وهي Rom فيها برنامج مهمته عملية الإقلاع أي البحث عن الoperating system ويقوم بعملية القراءة والكتابة الفعلية على الأقراص .
- Buffer : هي منطقة في الذاكرة الرئيسية تتوسط بين القرص والبرنامج ، فإذا أردنا برنامج ما يكتب في القرص فإنه يكتب في الBuffer وبعد ذلك يقوم نظام التشغيل بأخذ البيانات من ال buffer ومن ثم كتابتها في القرص ، والعكس صحيح فإذا أراد برنامج أن يقرأ بيانات من القرص فإن نظام التشغيل يقرأ البيانات من القرص ثم يكتبها في الBuffer وبعد ذلك يقوم البرنامج بقراءتها من الخازنة الوسيطة .
- **أولاً الكتابة على القرص :-**

1. يطلب البرنامج من الbois الكتابة على القرص .
2. يبحث الbois عن مكان خالي في الbuffer ثم يرجع العنوان إلى البرنامج .
3. يكتب البرنامج البيانات في شكل block في العنوان المحدد من قبل الbois .
4. يأخذ الbois البيانات من الbuffer ويخزنها في القرص .

ثانياً القراءة من القرص :

1- يطلب البرنامج من الbios القراءة من القرص .
2- إذا كانت البيانات في الbuffer فاذهب إلى الخطوة رقم (4) .
3- يجلب الbios البيانات من القرص ثم يخزنها في الbuffer .
4- يرسل الbios العناوين من الbuffer إلى البرنامج .
5- يقرأ البيانات من الbuffer .

❖ ملاحظة هامة :

في حالة أن الbuffer كلها مليئة يقوم الbios بإختبار إحدى الblock لعوامل معينة ثم يخزنها في القرص حتى يمكن الإستفادة من مكانها لمكان شاغر وهذا مُيعرف بعملية ال"paging" ، معظم خطوات القراءة والكتابة تتم بشفافية كاملة عن المستخدم (transparent) .

تبديل التجميعات :

إذا طلب البرنامج من الBios عملية إدخال أو إخراج ، ففي كلا الحالتين فإن الBios ستحتاج إلى مكان فارغ في الbuffer .

فماهو الحال إذا وجد الBios أن الخازنة الوسيطة ممتلئة والإفراغ فيها ؟

في هذه الحالة يقوم الBios بإحالة أحد هذه التجميعات إلى القرص حتى يستخدم مكانه ، تسمى هذه العملية بال(buffer replacement) وتُسمى بلغة نظام التشغيل ال(paging) و هناك عدة سياسات يستخدمها مدير الخازنة الوسيطة (buffer manger) في عملية إزاحة تجميعات الbuffer ، وأشهر هذه السياسات هو ما يسمى بـ(Least Recently Used (LRU)) ، هنا يتم اختبار أقل التجميعات استخداماً من قبل البرنامج حتى يتم إزاحتها من الخازنة الوسيطة لكي يستخدم مكانها .

فإذا كانت هذه التجميعية قد أجريت عليها تعديلات منذ جلبها من القرص فإنها تكتب مرة ثانية في القرص ، أمّا إذا لم تكن قد أجريت عليها تعديلات فإنها تزال فقط من الbuffer حتى يستخدم مكانها .

الأقراص المغناطيسية Magnetic Disks :

البيانات يجب تخزينها أو حفظها بصورة دائمة في إحدى وسائط التخزين الثانوية وذلك للآتي :-

1- البيانات قد تكون كثيرة بحيث لاتسعها الذاكرة الرئيسية .

2- احتمال ضياع البيانات في mm اكبر من ضياعها في القرص .

3- تكلفة الوسط الثانوي أقل من تكلفة الذاكرة الرئيسية .

تعتبر الأقراص المغناطيسية أكثر الوسائط الثانوية استخداما في الحاسبات ، حيث أن البيانات بها تكون متاحة On-line مقارنة بوسائط التخزين الثانوية الأخرى والتي تكون Off-line بمعنى أنها تكون غير جاهزة وتحتاج لعمل يدوي أو ميكانيكي لتوصيلها بالنظام ، أهم ما يُميز الأقراص الممغنطة :

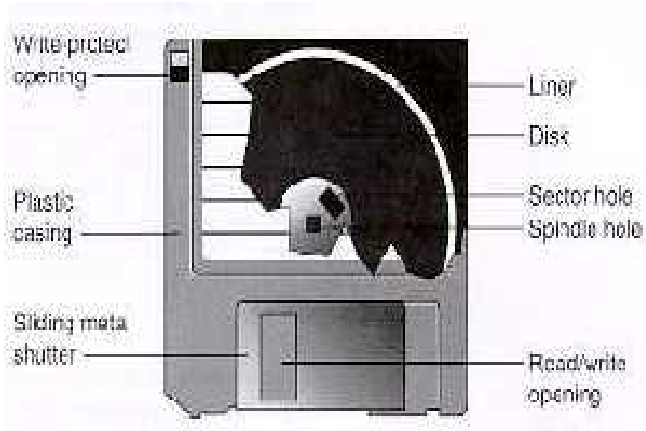
- سعة التخزين الكبيرة جداً بالنسبة إلى الأشرطة المغناطيسية .
- سرعة الأداء في استرجاع البيانات .
- الحجم الصغير والوزن الخفيف .

ومن أنواع الأقراص الممغنطة الأقراص المرنة والأقراص الصلبة .

أ- القرص المرن :

تتصف الأقراص المرنة بأنها ذات طاقة تخزينية صغيرة ،كما أن زمن الوصول إلى المعلومات طويل، وتتم قراءة القرص بواسطة السواقة عن طريق رأس القراءة والكتابة وتختلف عن الأقراص الصلبة كونها مصنوعة من مادة بلاستيكية بدلا من المعدن الموجود في القرص الصلب ،لذلك فهي أبطأ في التعامل والوصول للمعلومات من القرص الصلب ،وتتميز بأنها معدة لنقل المعلومات ويعتبر مُشغل الأقراص المرنة هو الأكثر عطباً لإحتوائه على أجزاء متحركة بعد استخدامه افترة طويلة .

كانت الأقراص المرنة شائعة الإستخدام في الثمانينات والتسعينات خاصة مع الحواسيب المنزلية كماكنتوش وحواسيب IBM المنزلية لتوزيع البرامج وتبادل البيانات وأخذ النسخ الإحتياطية وكانت تستخدم لتخزين نظام التشغيل وبرامج الحاسوب المنزلي وذلك قبل اختراع الأقراص الصلبة ، نهاية التسعينات ومع انتشار الإنترنت بشكل واسع بدأ منتجوا البرامج باستخدام الأقراص الليزرية ، واستخدام أجهزة تخزين ونقل بيانات ذات سعات أكبر، مما أدى إلى تضاؤل استخدام الأقراص المرنة .



شكل(10)

تصنع هذه الأقراص المرنة من قاعدة بلاستيكية على هيئة قرص ، وتتميز هذه الأقراص بمرونتها وخفة وزنها وصغر حجمها ، وتوضع الأقراص المرنة داخل غلاف مربع من الورق المقوى ، وهناك أنواع أخرى غلافها يكون من البلاستيك الصلب ، ويقوم الغلاف بحماية الأقراص المرنة ، كما يوجد بكل غلاف حافظ للقرص تساعد على توضيح بداية مسار التسجيل وأيضاً فتحة دائرية بمركز القرص تساهم في تحديد وضع القرص في موضعه داخل وحدة تشغيل الأقراص .

منذ عام 1968 بدأ استعمال القرص المرن ذو الحجم 5 1/4 ، وفي ذلك الوقت لم يكن بالإمكان وضع سوى 140 byte على ذلك القرص ، ولكن هذه الكمية كبيرة آنذاك ، والآن يمكن تخزين مايزيد عن 1.2mbyte من المعلومات على سطحه ومع ظهور الحواسيب من نوع Macingtosh[2] شاع استعمال الأقراص ذو الحجم 31/2 بوصة ، كما أن سعتها تكون حوالي 2.8mbyte وهي الأكثر متانة .

التركيب الداخلي العام لمُشغل الأقراص المرنة :

يعتبر مشغل الأقراص المرنة وحدة مستقلة خفيفة الوزن يبلغ عرضها حوالي 6 بوصات وبعمق 8 بوصات وتتوفر في ارتفاعين احدهما ارتفاع كامل full height وهو 3 بوصات ، والآخر ارتفاع نصفي half height وهو 1.5 بوصة .

❖ **ملحوظة :**

* إن استعمال القرص لمدة طويلة وفجأة لم يكن بالإمكان قراءته فالمشكلة ليست في مشغل الأقراص بل في عمر القرص الذي يكون له عمر محدد (يُقدر بثلاث سنوات) قد انتهى فالأقراص لايمكن أن تقدم الخدمة إلى مالانهاية ، حتى وإن كان استعماله بسيط .

* وجود قرصين مرنين ملتصقين مع بعض لا يؤثر على محتواهما من البيانات، لصغر تأثير المحتوى المغناطيسي فيهما.

ب- القرص الصلب :-

عندما أنزلت شركة IBM في عام 1983 القرص الصلب الضخم بسعة 10mbyte مع حواسيبها IBM –PC\XT بدا الأمر غير معتاد وغريباً في مجال الحواسيب الشخصية
القرص الصلب جزء من مكونات الحاسوب الأساسية وهو المسئول عن التخزين الطويل الأمد للمعلومات وهو عبارة عن أقراص معدنية مطلية بماده مغناطيسية موضوعه داخل علبه محكمه الإغلاق مفرغه من الهواء. تصل السعه التخزينية اليوم الىGB100 . ويمتاز بالسرعة عاليه تصل الى حوالي MS 10 مليون جزء من الثانية ،ويجب ان نأخذ في الاعتبار عده نقاط هامه:-
يتكون القرص من مجموعة الـpllaters وهي شرائح مغناطيسية دائرية رقيقة مصنوعة من مادة صلبة كما في الأقراص الصلبة hard disk أو تكون مصنوعة من مادة بلاستيكية كما في الأقراص المرنة floppy disk .
يتم تخزين البيانات على أسطح هذه الشرائح فإذا كان التخزين على وجه واحد فقط من وجهي الشريحة يطلق على القرص single sided disk .
بينما ال double sided disk يتيح إمكانية التخزين على وجهي الشريحة .
بالنسبة للقرص الصلب توجد منظومة من الشرائح disk pack تضم عدة شرائح يُقسم كل سطح من هذه الشرائح منطقياً إلى مجموعة من المسارات الدائرية tracks ، هذه المسارات بدورها تنقسم إلى وحدات صغيرة تُسمى sectors or blocks .
اعتماداً على نوع القرص تختلف أحجام الـsector وعدد المسارات tracks بين مئات أو إلى عدة آلاف من الـtracks في الشريحة الواحدة بينما يتم تقسيم المسار track إلى عدد من الـsector بواسطة نظام التشغيل أثناء تهيئة القرص formatting ، لذا فإن حجم الـsector ثابت ولا يُمكن تغييره ، وغالباً مايتراوح الـsector الواحد بين 32kb إلى 4096kb .
هذه الـsectors تفصل عن بعضها بمساحات محددة تسمى بالـinterblock gaps ، هذه المساحات تضم بيانات تحكم خاصة تكتب بها أثناء عملية التهيئة initialization
هذه البيانات تستخدم لتحديد موضع الـsector في الـtrack .
هناك تحسن كبير في صناعة الأقراص الصلبة من ناحية السعة التخزينية ومن ناحية السرعة في استرجاع البيانات أن أسعارها في انخفاض مستمر .

الأقراص (الأطباق الدائرية) :
هي مجموعة من الأقراص المتصلبة الدائرية الشكل المصنوعة من المعدن (الألمونيم) أو البلاستيك ، وأما الحديثة فتُصنع من الزجاج المقوى بالسيراميك الذي يُعتبر الأفضل من حيث مقاومة الإرتفاع في درجات الحرارة ،وجهي كل قرص مغطى بطبقة من أكسيد الحديد أو أي مادة أخرى قابلة للمغنطة ،كل الأقراص مثبتة من مركزها على محور دوران يعمل على تدوير كل الأقراص بنفس السرعة .
طلي الأقراص :

تطلى الأقراص بمواد لها خاصية حفظ الشحنة المتكونة عليها أو القراءة لحفظ البيانات وهذه المواد كأي مادة صلبة عندما تطحن تصبح حبوب صغيرة جداً ، وهذه الحبوب هي التي تخزن فيها الشحنة بواقع بت واحد لكل حبة لذلك فهي صغيرة للغاية حتى يمكن تخزين عدد كبير من البيانات في أصغر مساحة ممكنة ، وهذه المادة المستعملة هي أكسيد الحديد مخلوط مع مادة صمغية ومادة أخرى شحمية لتكون مزيج يمكنه الإلتصاق بسطح القرص الصلب .

الدوائر الإلكترونية :

الدوائر الإلكترونية تقوم بترجمة الأوامر الصادرة من الكمبيوتر ثم تقوم على ضوء تلك الأوامر بتحريك رؤوس القراءة والكتابة إلى مكان معين من الأقراص مما يسمح لرؤوس القراءة والكتابة بقراءة أو كتابة البيانات المطلوبة .

وسنتعرف على أجزاء القرص الصلب ففي الشكل التالي صورة حقيقية لـhard disk من الداخل وتشمل أقراص التخزين .

لوحة التحكم(logic bord) :

وهي اللوحة الإلكترونية التي تتحكم بالقرص الصلب (الرؤوس و المحرك) وتقوم بعمليات القراءة والكتابة من وإلى القرص .



شكل(11)

حيث أن :

1- هي رؤوس للقراءة و الكتابة .

2- مُحرك رؤوس القراءة والكتابة .

3-المحور المشترك لرؤوس القراءة والكتابة . 4- محرك الأقراص .

رؤوس القراءة والكتابة :

تُثبت رؤوس القراءة والكتابة على ذراع أفقي يمتد على من السطحين العلوي والسفلي لكل واحدة من الأقراص الدائرية ، الذراع الأفقي يتحرك ذهاباً وإياباً بين مركز الأقراص وحافتها الخارجية وبسرع كبيرة وهذه الحركة مع حركة دوران الأقراص الدائرية تسمح لرؤوس القراءة والكتابة بالوصول إلى أي نقطة على سطح الأقراص .

مُحرك رؤوس القراءة والكتابة (Actuator) :

يقوم هذا المحرك مع الأجهزة الإلكترونية الخاصة به بتحريك الرؤوس للمكان المطلوب من القرص حتى يمكن استخدام كافة مساحة القرص في تخزين البيانات ، ونظرا لأن المسافة بين البتات صغيرة جداً فإن دقة المحرك في تحريك الرأس إلى المكان المطلوب بالضبط من الأمور الأكثر أهمية ، وحتى لايُخطئ محرك رؤوس القراءة والكتابة في مكان بت ما كان لابد من أساليب للتأكد من كون رأس القراءة في المكان الصحيح ، أحد هذه الأساليب هي تلقي المحرك معلومات عن مكان رأس القراءة عن طريق أنظمة إلكترونية خاصة كما يوجد آلية تصحيح كما سيأتي ذكره لاحقاً .

مُحرك الأقراص (spindle motor) :

هو عبارة عن محرك يقوم بتحريك الأقراص بسرعة معينة تُقاس بوحدة (RBM دورة في الدقيقة) وتدور الأقراص بسرعة دوران تتراوح عادة مابين 4500 و 5400 دورة في الدقيقة وقد تصل إلى 10 000 في الدقيقة أو أكثر حسب نوع القرص ، وكلما كان معدل دوران المحرك أسرع كان ذلك أفضل في سرعة الإستجابة لإوامر رؤوس القراءة والكتابة .

كيف تتم القراءة و الكتابة على الأقراص الصلبة ؟

تخزين البيانات على القرص أشبه بتخزين الكتب في المكتبة، سنجد أن المكتبة مقسمه إلى عدة أجزاء وكل جزء يحوي عدة أرفف(رفوف)، وكذلك القرص الصلب فهو مقسم لعدة كلسترات وكل كلستر Cluster يحوي عدد معين من القطاعات...

يتم تخزين البيانات على شكل شحنات كهرومغناطيسية وتختلف قيمة هذه الشحنات لتعبر عن البيانات المختلفة.

عملية الكتابة:

وهي مغنطة سطح القرص بشحنه كهرومغناطيسية ذات قيمه معينه.. وتتحكم الدوائر الالكترونية الموجودة في اللوحة الالكترونية المطبوعة المثبتة بجسم محرك الأقراص الصلبه بهذه العملية، من حيث تحديد المكان الذي سيتم الكتابة عليه وقيمة الشحنة الخاصة بكل بيان.

عملية الكتابة تمر بالمراحل التالية:

- يصدر المعالج أوامره لمحرك الأقراص الصلبه بتخزين بيانات معينه.

- يستقبل المحرك هذه البيانات عبر البينية المتصل بها مع اللوحة الأم سواء كانت (EIDE) أو (SCS).

- يبحث محرك الأقراص في جدول الملفات عن أماكن خاليه.

- يوجه الإبرة إلى المكان الخالي ويعطيها البيانات على شكل شحنات كهرومغناطيسية.

- تقوم الإبرة بكتابة (مغنطة سطح القرص) هذه البيانات بنفس التوالي الذي استقبلته من دوائر القرص الالكترونية.

عملية القراءة:

هي قراءة الإبرة لقيمة الشحنة الكهرومغناطيسية وتقوم الدوائر الالكترونية مره أخرى بترجمة هذه الشحنة وتحويلها إلى بيانات.

عملية القراءة تمر بالمراحل التالية:

- عندما يطلب المعالج بيانات مخزنه بالفعل على القرص الصلب، فإن القرص يبحث في جدول الملفات عن هذه البيانات وعنوانها على سطح القرص. (العنوان كما سبق وذكرنا يتضمن رقم الاسطوانة والرأس والقطاع CHS).

- يوجه محرك الأقراص الإبرة إلى مكان البيانات لتقوم بالقراءة.

- تتم القراءة (بقراءة قيم الشحنات الكهرومغناطيسية وإرسالها إلى الدوائر الالكترونية الخاصة بمحرك الأقراص لترجمتها وإرسالها للمعالج).

عملية الحذف:

- عند قيام المستخدم بحذف ملف معين من على القرص الصلب فان المعالج يصدر أمر الحذف إلى محرك الأقراص الصلبة.

- يقوم محرك الأقراص بتنفيذ أمر الحذف.

- الخطوة المنطقية التالية هي توجه الإبرة إلى مكان البيانات وحذفها بإزالة الشحنات الكهرومغناطيسية التي تعبر عن البيانات، ولكن...

وجد مصمموا الأقراص أن توجيه الإبرة إلى مكان البيانات (لمسح) الشحنات الكهرومغناطيسية تمثل عبئاً إضافياً على المحرك كما تمثل إهدارا للوقت، لذا فقد تم إتباع الطريقة التالية:
يقوم المحرك بتعيين مكان البيانات في جدول الملفات ثم يعرف الجدول بان هذا المكان (أصبح خاليا... وبالتالي عند تنفيذ أمر كتابه فانه من الممكن الكتابة في هذا المكان (فوق البيانات السابقة).

بمعنى آخر: عند حذف الملفات من على القرص الصلب فانه يتم حذفها (وهميا) بحيث تظل البيانات موجودة بينما لا يمكن للحاسب التعرف عليها.

ومن هنا ظهرت برامج (استرجاع البيانات بعد حذفها) والتي تقوم بتوجيه الابره لقراءة (الأماكن الخالية) وترجمة الشحنات الموجودة عليها لمعرفة ما إذا كان هناك بيانات (سليمة) يمكن استعادتها أم لا.

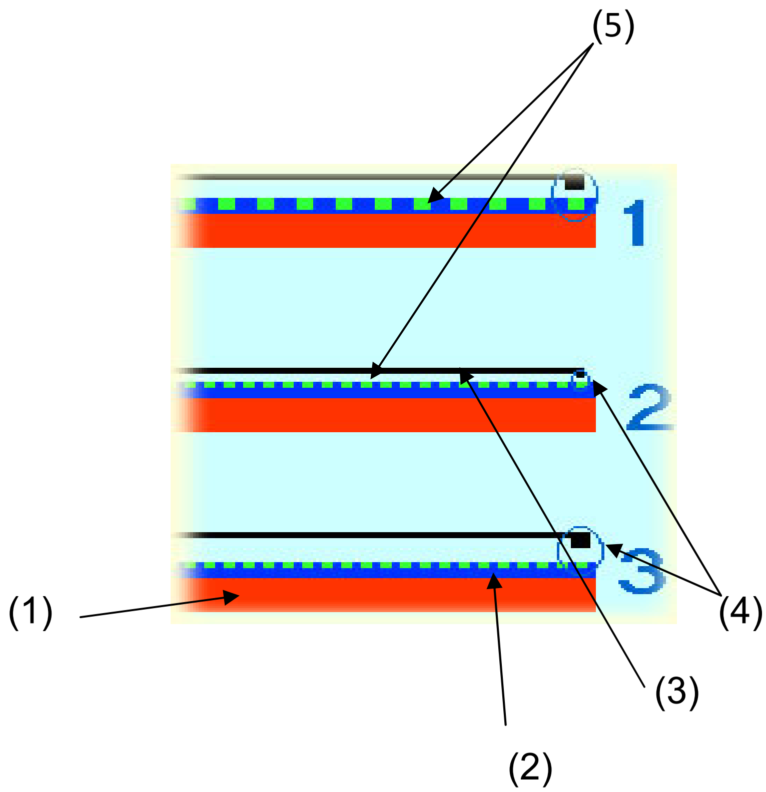
ونظريا: فانه باستطاعتك استعادة جميع البيانات المحذوفة ما لم يتم كتابة بيانات أخرى على سطح القرص. ولهذا السبب أيضا ظهرت برامج (مسح) البيانات نهائيا مثل برنامج Erase الشهير.. ووظيفة لبرنامج توجيه الإبرة (لإزالة) الشحنات الموجودة على سطح القرص لمسحها نهائيا. وهذا النوع من البرامج يستخدم بكثرة في الجهات الأمنية لمنع إمكانية استعادة ملفات حساسة تم حذفها.

* إن عدد البتات التي يمكن تسجيلها على المسارات الخارجية للقرص أكبر من تلك التي يمكن تسجيلها على المسارات الداخلية بسبب شكله الدائري لذا فإن رأس القراءة والكتابة يجب أن يقرأ أو يكتب بمعدل أسرع في الطرف الخارجي عن الداخلي .

* كما أن رؤوس القراءة والكتابة كلما كانت أصغر حجماً كان بإمكانها التسجيل في حقول بتات أصغر وبالتالي الحصول على كثافة أعلى للبيانات وأيضاً يُمكن للرأس الأصغر الإقتراب من سطح القرص أكثر وأكثر من دون الإحتكاك به ، والإقتراب من سطح القرص يعني إمكانية تخزين بيانات أكثر .

* عندما يكون رأس القراءة والكتابة بعيداً عن سطح القرص فإن المجال المغناطيسي يجب أن يكون كبيراً حتى يُمكنه التأثير على سطح القرص وإذا كان كبيراً فإنه يُمكن أن يؤثر على البتات التي بجانب البت المراد التأثير عليه وهكذا يُمكن أن يحدث الخطأ في القراءة والكتابة .

الشكل(12) التالي ثلاثة أشكال لأقراص صلبة :



حيث يمثل :

(1) سطح القرص .
(2) المادة المغناطيسية التي تخزن البيانات .
(3) رأس القراءة والكتابة .
(4) المجال المغناطيسي الذي يحيط برأس القراءة والكتابة .
(5) مواقع تخزين البيانات .

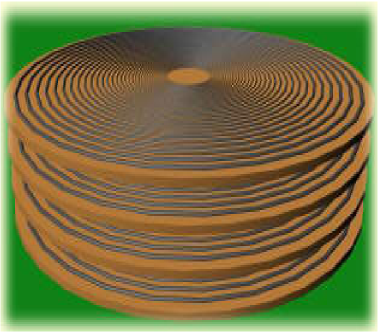
عند المقارنة بين الشكل الأول والثاني نجد أن كثافة القرص الأول أقل من الثاني وذلك لأن عدد البتات أكبر في القرص الثاني من القرص الأول ، ورأس الكتابة أقرب وأصغر والمجال المغناطيسي أصغر .

- **السعة size** : فكلما زادت سعة القرص الصلب التخزينية كلما زادت الطاقة التخزينية للبرامج والمعلومات التي يحتفظ بها الكمبيوتر ونلاحظ ان هناك اقراص صلبة تبدأ سعتها من ١ الى ١٦٠ جيجا بيت.

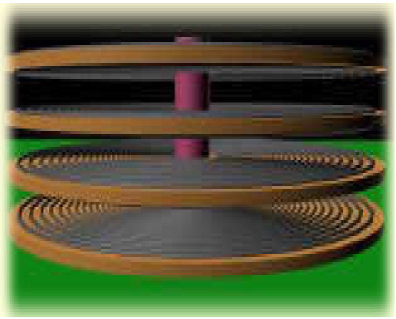
• معدل نقل البيانات transfer rat : وهي كمية نقل البيانات داخل الحاسب في الثانية الواحدة.

• معدل سرعة الدوران disk rotation : والمقصود بها عدد المرات التى يدورها مشغل القرص الصلب في الدقيقة وتتراوح بين ٣٦٠٠ ، ٧٢٠٠ دورة في الدقيقة الواحدة وكلما زاد عدد هذه الدورات كلما زادت كفاءة الكمبيوتر .

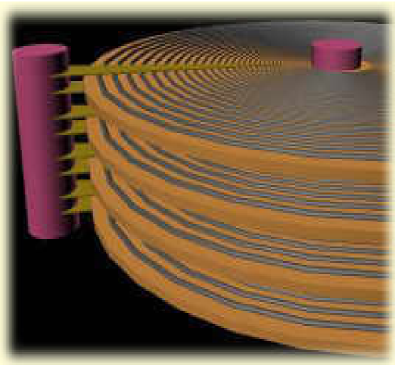
فيما يلي مناظر إفتراضية لأشكال الأقراص من جهات مختلفة :



- منظر افتراضى لشكل الأقراص(platters) الدائرية والتى يتم التسجيل على كل من وجهيها وكلما زاد عدد الأقراص وكثافة البيانات التي عليها كلما زادت قدرة القرص الصلب على تخزين البيانات ، تسمى كمية البيانات التي يمكن تخزينها في مساحة معينة من سطح القرص areal density، وأكثر الوحدات استخداماً هي الميجابايت لكل إنش مربع (MB/square inch) .



- منظر افتراضي لشكل محور الأقراص الذي يمكنها من الدوران حول محورها معاً .

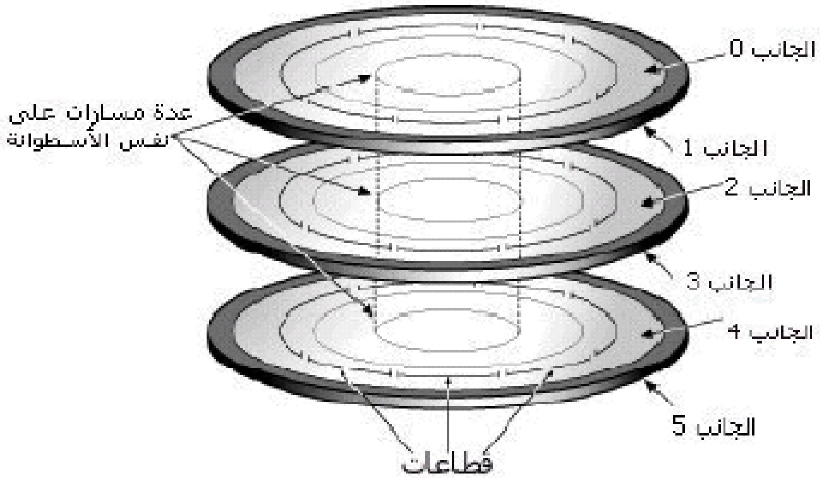


- منظر افتراضي لشكل محور الأقراص ورؤوس القراءة والكتابة (رأس على كل سطح من السطوح) ولأن المسافة بين القرص ورأس الكتابة صغير جداً (أجزاء من الألف من الإنش) فإن هذه الأقراص يجب أن تكون مستوية تماماً بحيث لا تلتمس مع الرأس أثناء العمل وإلا تعطل القرص بسبب ذلك.

ورؤس القراءة والكتابة تتحرك كلها معاً لأنها على محرك واحد وقاعدة واحدة ، ورأس القراءة والكتابة محمول على ذراع مرن قليلاً مما يمكنه من ملامسة القرص أو الارتفاع عنه قليلاً ، فعندما يكون القرص واقفاً فإن رأس القراءة والكتابة يكون ملامس لسطح القرص و عندما يبدأ القرص في الدوران فإن تيار الهواء الناتج من الدوران يبعد رأس القراءة والكتابة عن سطح القرص قليلاً (المسافة قليلة إلى حد أجزاء من المليون من الإنش) بحيث لا يحدث تلامس بينهما أثناء العمل ، وعندما يود القرص الصلب إيقاف الدوران فإنه يحرك الرأس لمكان آمن

من القرص يسمى منطقة الهبوط (landing zone) حيث يمكن بعدها إيقاف دوران القرص والسماح برأس القراءة والكتابة بملامسة سطح القرص حيث أن منطقة الهبوط خالية من البيانات فهي مخصصة فقط لهبوط الرأس عليها ، ليس هذا فحسب بل يتم أيضاً "ربط" الرؤوس في منطقة الهبوط حتى لا يتحرك الرأس مع ارتجاج القرص الصلب وهذه العملية تتم أوتوماتيكياً في الأقراص الجديدة أما القديمة جداً فقد كانت تستلزم برنامج خاص لعمل ذلك .

خصائص الأقراص الفيزيائيهphysical Characteristics of Disks:-



شكل(14)

Disk Controller:-

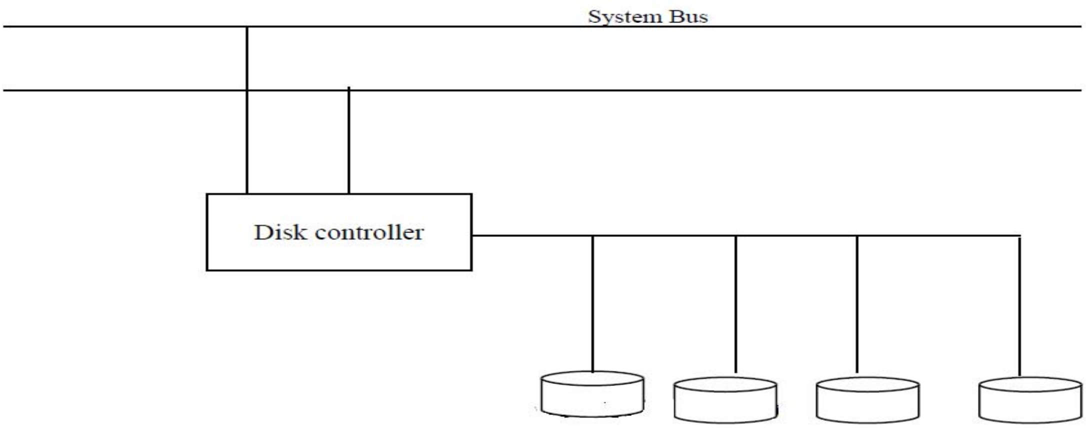
هذه الوحدة تعتبر جزء من الDIsks وهي تتحكم فيه وتشكل الواجهةinterface بينه وبين نظام الكمبيوتر .واحدى انواع الواجهات القياسية والمستخدمة حالياً مع الحاسبات الشخصيه تسمى بSCSI (small computer storage interface)

تتلقى الController أوامر القراءة والكتابة المكتوبة بلغات المستوى العالي لتتخذ بناء عليها الحركة المناسبة لوضع الذراع الArm وبالتالي وضع راس القراءة والكتابة في الموضع المطلوب لحدوث عملية القراءة او الكتابة.

كذلك تقوم الController بإلحاق وحدة تسمى Checksum بكلSector يتم كتابة عدد البيانات المخزنة بها.

وعند قراءة هذه البيانات يتم حساب حجم البيانات التي تمت قراءتها ويقارن هذا الحجم بقيمة الChecksum المخزنة في كلSector فاذا لم تتطابق القيمتان فان هناك خطأError, في هذه الحالة تكرر الController المحاولة عدة مرات ، فاذا إستمر الخطأ في الحدوث تنبه الcontroller نظام التشغيل الى وجود مشكلة في عملية القراءة Read Failure .

فيما يلي شكل يوضح نظام فرعي للقرص :



شكل(15).

العنونة ونقل البيانات : -

عند نقل البيانات من و الى القرص يتم التعامل بوحدة تسمى الDisk Block وهو يمكن ان يكونSector واحد أو مجموعة من الSectors المتتالية في نفسTrack حيث يتم تحديد عنوان الBlock في القرص والوصول اليه مباشرة .وعنوان الBlock يتكون من رقم السطح الPlatter ، رقم المسار الTrack (داخل السطح)، ورقم الBlock (دا خل ال Track).

وتقوم وحدةI/O Hardware الموجودة بالقرص نفسه بتحديد هذا العنوان.

ذاكرة الBuffer الخاصة بالقرص عبارة عن جزء من الذاكرة الرئيسية يتم نقل البيانات بينها وبين القرص فعند عملية القراءة يتم نقل البيانات بينها وبين القرص . فعند عملية القراءة يتم نسخ البيانات من الBlock الى الBuffer وعند الكتابة يتم نقل محتويات الBuffer الى القرص الصلب . احيانا يتم نقل البيانات في شكل مجموعة من الBlock كوحدة واحدة تسمى الCluster في هذه الحالة يجب زيادة حجم الBuffer حتى تسع كل الCluster .

تنتقل البيانات بين القرص والذاكرة الرئيسية في شكلBlocks ويقوم ال File System Manager بتحويل عنوان الBlock لرقم الPlatter والCylinder والTrack وال Block المطلوب التعامل معه.

مقاييس أداء القراص(Performance Measure of Disks) :

المقاييس الأساسية لأداء القرص هي:

1. Capacity السعة :-

ونجد أن الحاسبات الحديثة توفر سعات تخزينية ضخمة تتيح تخزين كميات كبيرة من البيانات

2. Access Timeزمن الوصول للبيانا ت: -

هو الزمن المطلوب لتحديد موضع الBlock المطلوب قراءته أو الكتابة به لتبدئ عملية نقل البيانات بين القرص والذاكرة الرئيسية ويتكون من:

(a) زمن البحث (Seek Times) :-

هو الزمن المطلوب لتحريك رأس القراءة والكتابة الى الTrack المطلوب في ال Movable –head disks أما في الFixed –head disks فهو الزمن المستغرق لتحديد الراس المطلوب الكترونيا Electronically switch to the appropriate head في ال Movable-head disks يختلف هذا الزمن على حسب موضع الراس الحالي والموضع المطلوب الإنتقال اليه وعادة ماتحدده الشركات المصنعة هذا الزمن كمتوسط Average seek Time بالملي ثانيه وهو عادة مايتراوح بين10 -٦٠ msec في الحاسبات الشخصية ومن8 - 9 في اجهزة الservers .

(b) (Rotational DelayTime rd):-

عندما يكون راس القراءة والكتابة في ال الtrack المطلوب يجب الإنتظار حتى تدور ال Platter لتصل لبداية الBlock المطلوب أسفل الراس هذا الزمن يمكن ان يكون صفر اذا كان الBlock المطلوب تحت الراس بمجرد وصول الراس للTrack المعني أو يمكن أن يكون زمن دورة كاملة للPlatter اذا كان ال Block المطلوب هو الBlock قبل ال Block الموجود حالياً تحت الرأس ، إذاً فهو في المتوسط زمن نصف دورة .
فإذا كانت سرعة دوران القرص = p دورة في الدقيقة(P=Rotation per minute(rp
إذاً يكون متوسط زمن الدوران(Average rotation delav (rd) :

$$rd = ((1/2) * (l/p)) \min \Rightarrow rd = (60 * 1000) \quad / 2p$$

أما بالنسبة للFixed –head disk يعتبر الSeek Time صغير لذا تكون الrd هي القيمة المؤثرة على سرعة القراءة والكتابة.

إذاً الAccess time هو مجموع الل seek time +rd.

3. Data Transfer Rate (Tr) أو Block Transfer Rate :-

هو معدل نقل البيانات من والى القرص بعد وصول الرأس لبداية الBlock المطلوب نقله .
هناك زمن مطلوب لنقل البيانات من الBlock الى الذاكرة الرئيسية وزمن نقل ال Block (Block Transfer time (btt) يعتمد على حجم الBlock ، وحجم الTrack وسرعة دوران القرص، وبهذا يكون الزمن الكلي المطلوب لتحديد موضع الBlock ونقل محتواه هو مجموع :

Seek time + Rotational delay + Block transfer rate

مثال:-

قرص صلب معدل نقل بياناته يساوي10 m sec وزمن البحث seek time هو 5 m sec إذا علمت أن قيمة الRotational delay time هي صفر، احسب الزمن اللازم للوصول الي الblock ونقل محتواه ؟

الحل:-

Access time = Seek time +Rotational delay +Block transfer rate

=> 5+0+10= 15 m sec

وعموماً الSeek Time والRotational delay زمنيهما أكبر من ال (Block Transfer Rate) حيث أن الزمن الأكبر للحركة الميكانيكية ، ولزيادة كفاءة وسرعة نقل البيانات توضع الBlock التي يراد التعامل معها (قراءة أو كتابة) في اسطوانة واحدةCylinder لتقليل الحركة الميكانيكية . وعلى كل حال يعتبر الزمن المطلوب للوصول الى هذه البيانات المحددة على القرص كبيرا مقارنة بزمن تشغيلها في الmain memory .

إذا كان معدل نقل البيانات (Transfer rate) لقرص يساوي tr Byte / msec وحجم الblock يساوي B Byte فإن :

Btt = B/tr msec

مثال:

قرص حجم الtrack به يساوي 50 KB وسرعة الدوران 3600 rpm يكون معدل نقل البيانات tr كما يلي :

Tr= ((50 * 1000) / (60 * 1000)) /3600 => 3000 byte/msec .

∴ **BTT** = B / 3000 mesc .

حيث B = حجم الblock بوحدة الByte .

لنقل بيانات من K Tracks ليس متتالية ولكنها في اسطوانة واحدة (cylinder) ونحتاج تقريباً

S+(K * (rd + Btt))

حيث تكون S مطلوبة فقط للوصول للblock الأول فقط .

وفي هذه الحالة نحتاج لحجم أكبر من الذاكرة الرئيسية ليكون الbuffer كافي للتعامل مع ال K blocks .

ولتقليل هذا الزمن توضع البيانات المتتالية ليس في اسطوانة واحدة فحسب بل توضع متتالية في نفس الTrack وهذا يحذف زمن الدوران (rd) عدا للblock الأول ، ولذا يكون الزمن المطلوب لنقل K blocks متتالية أي :

$$S + rd + (K * Btt)$$

ولنكون أكثر دقة في حساب زمن نقل الblock نأخذ في الحسبان المساحات الفارغة (Interblock gaps) ، والتي هي عبارة عن مساحات داخل الblock تحوي بيانات التحكم اللازمة لتحديد موضع الblock في الTrack وعادة يقدم منتج تلأقراص قيمة تُعرف بال(Bulk Transfer rate (btr)) ، والتي تأخذ في الحسبان هذه المساحات .

فإذا كان حجم ال G=gap فإن :

$$\mathbf{Btr} = ((B / (B + G)) * tr)$$

حيث B = حجم الblock بوحدة الByte .

الbtr هو معدل نقل البيانات الحقيقي حيث يمر الرأس على كل الbytes في الtrack أثناء دوران القرص بما في ذلك الInterblock gaps عند استعمال الbtr يكون الزمن المطلوب لنقل البيانات الحقيقية في الblock الواحد تساوي B / btr .

4. Reliability الإعتمادية :-

يتم قياس الReliability لقرص بمتوسط زمن تعطله عن العمل (mean time of failure is a measure of reliability of the disk)

وهو متوسط الفترة الزمنية التي يتوقع أن يعمل فيها القرص باستمرار دون توقف (متوسط زمن السقوط) .

RAID Technology:-

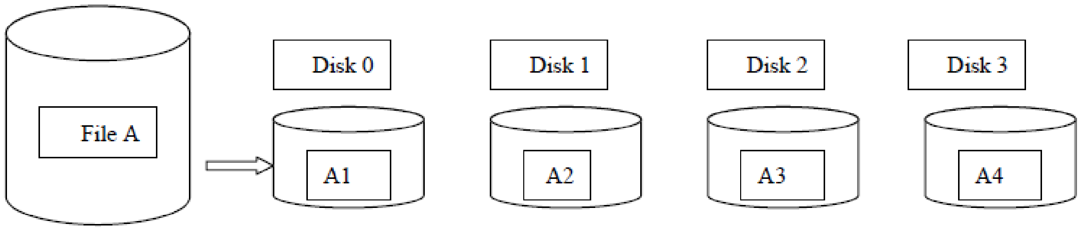
مع زيادة أحجام وسرعات الذاكرة الرئيسية والمعالجات كان لابد من زيادة أحجام وسرعات وحدات التخزين الثانوية لمقابلة هذه الزيادة ، ولمقابلة احتياجات التطبيقات الجديدة وكان من أميز التطورات في وحدات التخزين الثانوية الRAID Technology .

RAID: (Redundant Arrays of Inexpensive Disk)

الحرف (I) أحيانا يرمز للIndependent كان السبب الأساسي للتفكير في الRAID هو أن الزيادة في تحسين أداء الDisk لم يكن بنفس سرعة الزيادة في تحسين أداء الذاكرة والمعالج . فكان الحل إنشاء منظومه (Array) من الأقراص المفصولة عن بعضها لتعمل كلها كقرص(منطقي)واحد ليكون عالي الكفاءة باستعمال مفهوم Data Striping والذي يحقق مفهوم العمل على التوازيparallelism.

عملية الData Striping هي عبارة عن توزيع البيانات على الأقراص لتعمل كلها كقرص واحد وسريع.

الشكل التالي يوضح كيفية توزيع البيانات لملف واحد وعلى عدد 4 أقراص :-



شكل(16)

عملية توزيع البيانات(Data Striping) تعمل على تحسين الأداء الكلي بالآتي: -

١.تتيح لأكثر من عملية قراءة او كتابة ان تعمل على التوازي مما يحسن عملية نقل البيانات الكلي.٠

٢. يقوم ال RAID بتوزيع العمل على كل الDisks.

٣. يمكن تحسين الإعتماديه reliability بتكرار كتابة البيانات على أقراص مختلفة.

تحسين الإعتمادية باستخدام الRAID Improving Reliability with RAID:-

بإستعمال منظومة أقراص Disks مكونة من n قرص، احتمال العطل nمرة احتمال تعطل القرص الواحد.

فمثلا لمنظومة مكونة من 100 قرص إذا كان العمر الإفتراضي للقرص 200.000 ساعة أي 22.8 سنة . يكون العمر الإفتراضي للمنظومة فقط 2000 ساعة أي 83.3 يوم وبفرض أن هناك نسخة واحدة من البيانات مقسمة على كل الأقراص.

إذاً الحل هو كتابة نسخة إضافية من البيانات، ولكن عيب هذا الحل هو المساحة الإضافية المطلوبة لهذه النسخة إضافة للعمليات الإضافية (I/O) اللازمة لكتابة البيانات أكثر من مرة.

إذاً فالتقنية الشائعة هي ال Mirroring or Shadowing حيث تكتب البيانات في قرصين متطابقين في نفس الوقت واللذان يعملان منطقيا كقرص منطقي واحد .عند قراءة البيانات تقرأ من أسرع القرصين وعند تعطل أحدهما تقرأ من الآخر حتى يتم إصلاح الأول.

تحسين الأداء بإستخدام الRAID in Improving Performance :-

توزيع البيانات على منظومة الأقراص يزيد من سرعة نقل البيانات . وبما ان البيانات تقرأ بواقع block واحد في اللحظة فانه يمكن تطبيق الdisk striping لأدنى مستوى بتقسيم ال byte الى ٨ ثنائيات (bits) وتوزيع هذه ال bits على 8 أقراص مختلفة حيث يكتب ال bit رقم j في القرص رقم j وبهذه تزيد كمية البيانات المقرؤة في اللحظة

(bit –level data striping) .

ويمكن تطبيق ال data striping على مستوى اعلى من الbit حيث يمكن تقسيم الملف الى blocks وكل blocks يكتب في قرص منفصل block level striping بهذه الحالة يمكن الوصول الى blocks مختلفة في نفس اللحظة مما يزيد كفاءة زمن الوصول الى البيانات. عموما كلما زاد عدد الأقراص زادت كفاءة المنظومة وايضا زاد احتمال حدوث العطل وبالتالي زادت الحوجة الى Mirroring .

RAID Organization and levels :

تقسم الRAID من ناحية تنظيمية الى أنواع أو مستويات levels يعتمد هذا التصنيف على عاملين أساسين وهما :-

* Granularity of data striping .

* Pattern used to compute redundant information .

في البداية كان التصنيف من المستوى الأول الى الخامس ومؤخرا أضيفت level6 و level0 .

level 0 : البيانات لا تتكرر ولهذا سرعة الكتابة وتعديل البيانات عالية لكن البيانات لاتكتب مرتين . اما سرعة القراءة فهي اقل مما عليه في المستوى الأول حيث تكتب البيانات مكررة لكنها تستعمل تقنيه mirrored disks حيث تكون في الأخيرة سرعة القراءة عالية وبحيث يتم تحسين الأداء بجدولة عمليات القراءة ويتم تنفيذ العملية التي تتطلب زمن بحث اقل.

Level 2 : تستعمل مايسمى بالmemory-style redundancy والتي توفر parity bits وبهذا فهي تحتاج لثلاثة اقراص اضافيه لكل اربعة اقراص اصلية، مقارنة باربعة اقراص مقابل اربعة اصلية في ١ level وهي ايضا تتيح امكانية اكتشاف وتعديل الخطأ.

Level 3: تستعمل قرص واحد لتخزين الparity واربعة اقراص لتخزين البيانات معتمدة على الcontroller لتحديد أي القراص قد تعطل.

Level 4 , level 5 : تستعمل الblock level data striping وفي level 5 توزع البيانات والparity information عبر كل الأقراص.

Level 6 : تطبق مايسمى بالP+Q redundancy حيث توجد

(2bit for redundant data are store for every 4 bits of data)

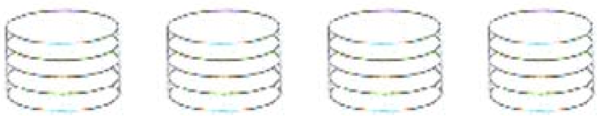
اعادة بناء البيانات اسهل في level 1 في حالة تعطل أي قرص . اما في بقية المستويات فهي تحتاج الى عملية معقدة للقراءة من كل الأقراص الأخرى . ولهذا فان level1 يستعمل في التطبيقات المهمة والحرجة أما level5 و level3 فهي تستعمل في البيانات كبيرة الحجم حيث تقدم level3 سرعة عالية لنقل البيانات.

إذاً فمصممي الRAID يجب أن يضعوا في الحسبان العديد من العوامل :

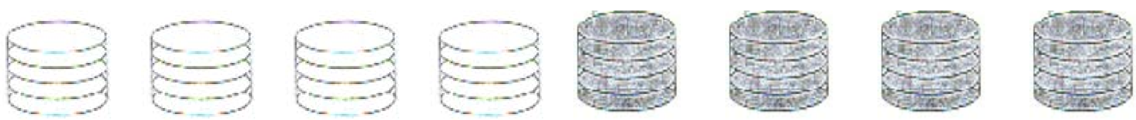
- RAID level .
- عدد الأقراص المطلوبة .
- طريقة الparity المستعملة .

بمناقشة الRAID Technology يتضح لنا التقدم في مجال تخزين البيانات وكيفية توظيف الأفكار نفسها يكون حسب التطبيقات .

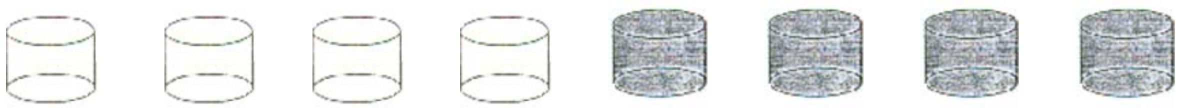
شكل يبين فيه مستويات الRAID :



NON-Redundant (RAID Level 0)



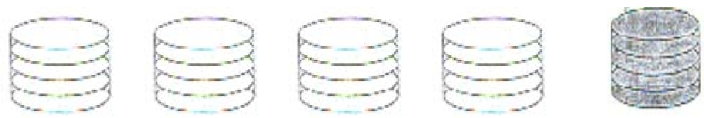
Mirrored (RAID Level 1)



Memory _Style ECC(RAID Level 2)



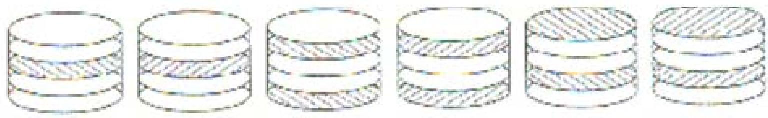
Bit-Interaved Parity(RAID Level 3)



Block –Interleaved Distribution-Parity(RAID Level 5)



Block –Interleaved Parity(RAID Level 5)



P+QRedundancy(RAID Level 6)

شكل (17) .

Buffering of Blocks:-

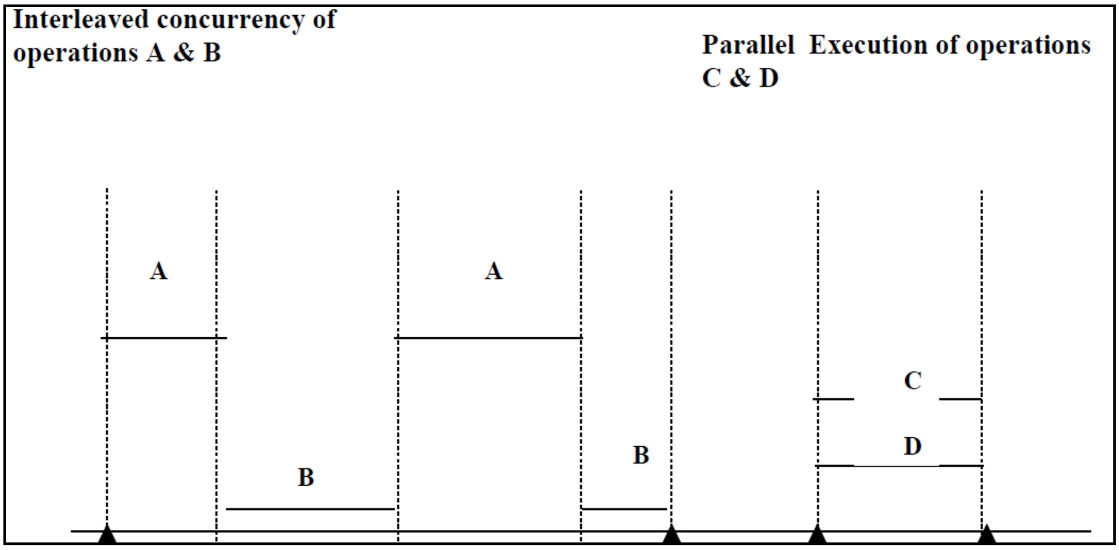
إذا كان هناك العديد من الBlocks يراد نقلها من القرص إلى الذاكرة الرئيسية وكلها معروفة العنوان .

ويمكن حجز أكثر من موضع في الذاكرة Several buffers لزيادة سرعة نقل البيانات .

أثناء قراءة أو كتابة One buffer يمكن للمعالج أن يشغل buffer أخرى ،وهذا يمكن بوجود Controller(disk 1/0 processor) والذي يعمل على التوازي وبمعزل من الCPU .

فيما يلي شكل يوضح كيف يمكن لمعالجين أن يعملان على التوازي ...

العمليتان A , B تعملان على التوازي ولكن بصورة Inter Leaved بينما العمليتان C , D تعملان على التوازي (in a paraller fashion) .

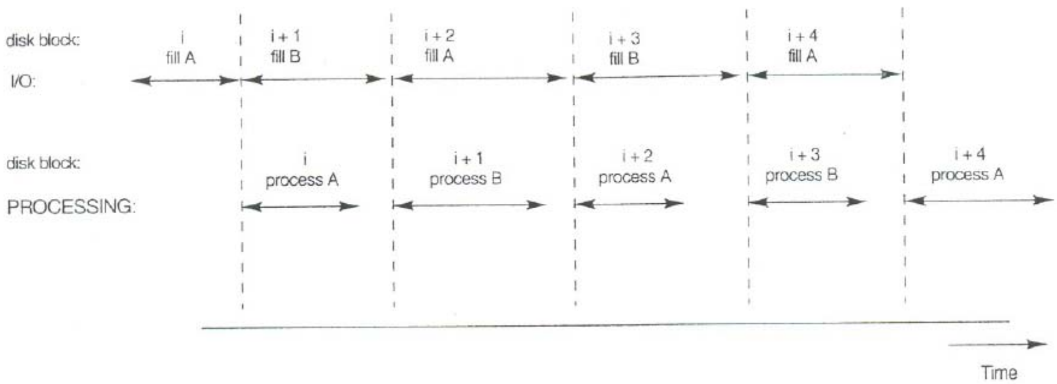


(Inter Leaved Concurrency Versus Parallel Execution) .

شكل(18)

بوجود معالج واحد يعمل على عدة عمليات يكون التشغيل على التوازي Paraller Executionغير ممكناً ،ولكن على كل حال يمكن للعمليات أن تعالج in an interladed way وفي حالة تشغيل العمليات على التوازيIn a paralllel fashion تكون عملية الbuffering مفيدة جداً إما لوجود معالج القرص الخاص disk 1/0 processor(controller) أو لوجود عدة CPU.

الشكل (19)يوضح كيف يمكن لعمليتي القراءة والمعالجة أن تعملان على التوازي عندما يكون زمن التشغيل على الذاكرة أقل من الزمن المطلوب لنقل الblocks التالي إلى الذاكرة .



Use of two buffer , A & B for reading from disk

يمكن لـCPUأن تبدأ عملها على الـblock بمجرد وصوله على الذاكرة الرئيسية وفي نفس الوقت تقوم(processor(controller بنقل disk 1/0 وقراءة block آخر على bufferأخرى.. هذه الطريقة تسمى double buffering وأيضاً تستعمل لكتابة فيض من الـblocks المتتالية من الذاكرة إلى disk . عملية الـdouble buffering تسمح بالقراءة والكتابة للبيانات الموجودة على الـblock متجاورة مما ينقص الـrotational & seek time delay لكل الـblocks عدا الـblock الاول .. زيادة على ذلك تكون البيانات جاهزة للتشغيل مما يقلل زمن الانتظار للبرنامج.

وضع السجلات الملف على القرص

Placing file records on disk

سنتحدث هنا عن السجلات ومفهومها وأنواعها ، وكذلك الملفات وبعض الطرق لوضع هذه السجلات على القرص .

السجلات record :

- هي نوع من أنواع البيانات المرتبة ، ولايشترط أن تكون العناصر المكونة لها من نفس النوع فهي مجموعة من البيانات المتعلقة ببعضها البعض ، وتختلف عنها في إمكانية إحتوائها على بيانات مختلفة في النوع .وتسمى عناصر البيانات الفردية الحقول field داخل السجل .
سندرس التطبيقات على السجلات والملفات بإستخدام لغة باسكال .

ونستخدم للتصريح عن السجلات طريقتين :

1.الطريقة الأولى هي صيغة الـ(Type) وهذه الطريقة الأكثر استخداماً :

```
type
recordName=record
filed1:dataType;
filed2:dataType;
filed3:dataType;

      ↓

filedn:dataType;
```

```
end;
```

```
var variableName:recordName;
```

أي في منطقة var نعرف المتغير من النوع recordname .

فمثلاً سجل الموظف يتكون من عدة حقول هي (الإسم : من نوع حرفي ، رقم الموظف: من نوع رقمي ......) وتكون كتابتها كالآتي :

```
type
employee=record
name :string;
no :integer;
salary :integer;
end;

var emp:employee;
```

يكون التعامل مع emp على أنه سجل وفيه ثلاثة حقول.

هذه الطريقة أعم لأنها تسمح بإدخال متغيرات سجلات إضافية .

2. الطريقة الثانية :

```
Var record name :Record

filed1:dataType;
filed2:dataType;
filed3:dataType;
        ↓
filedn:dataType;
End;
```

مثال لنفس سجل الموظف السابق :

```
Var employee :Record

name: string;

no :integer;

Salary : real;

End;
```

تشغيل السجل Record Processing :

اذا اردنا ان نحدد لسجل معين قيمه ما لسجل آخر ، فإنه لابد من أن تكون السجلات من نفس النوع ،فمثلاً:

```
VAR: old customer, New customer: customer;
Begin ;
New Customer: =Old Customer
END ;
```

وهذا يتم نقل جميع بيانات السجل Old customer إلى New customer .

ومن الممكن ايضاً نقل هذه البيانات من سجل لآخ عن طريق نقل حقل حقل .... فمثلاً

```
New customer. name : = old ustomer.name;
New customer.no   : =oldcustomer .no;
New customer .salary :oldcustomer.salary;
```

تشغيل السجلات عملياً يحتاج إلى هياكل بيانات أكبر منها لكي تحتوي السجلات ، فمثلاً لدينا ملف للزبائن أو منظومة تحتوي عدداً من الزبائن فهنا نحتاج لتعريف منظومة من نوع array عناصرها عبارة عن سجلات وبهذا يمكننا الوصول إلى السجل الموجود داخل الarray ، فمصفوفة مخزن بداخلها 100 سجل لزبائن تكون صياغتها كما يلي :

```
VAR Cust : Array [1..100] of customer ;
```

في الخطوة القادمة نقلنا بيانات السجل 35 بكامل بيانات حقوله إلى السجل 20 .

```
Cust [20] : = cust [35];
```

وهنا نقلنا بيانات حقل من سجل إلى الحقل المقابل له في سجل آخر بطريقة تسمى الdot...

```
Cust [20].no:=cust [35 ] .no;
```

وهناك طريقة أخرى لنقل البيانات أكثر من حقل من سجل لآخر وذلك بإستخدام with ، فبدلاً من تكرار إسم السجل لكل حقل وهذا ممل خاصة إن كانت الحقول كثيرة فإنه يمكن التعامل مع الحقول مباشرة بإستخدام الwith ، فمثلاً بدلاً من أن يكتب ...

```
New customer.name:='ali';
New customer.no:=1;
New customer.salary:=3500;
```

يمكن أن نكتب :

```
WITH Newcustomer DO
Begin
Custno:=16 ;
Custtype:='A ';
Custbalance:=315.6 ;
```

أنواع السجلات :

- تختلف أنواع السجلات بحسب أحجامها فهنالك سجلات ذات أحجام متساوية أي (Fixe Length Record) ، بمعنى أن هذا الfile كل السجلات لديها نفس الحجم من نوع البيانات أي أطوال السجلات محددة .

ولكن إن كان أطوال السجلات غير محددة فإن السجل يكون من نوع (Variable length Record) ، والحالات التي تكون فيها السجلات غير محددة أحد الآتي :

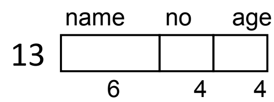
- إذا وجد حقل أو أكثر من حقل أحجامها مختلفة ، فمثلاً حقل الإسم يمكن أن يختلف من سجل لآخر.
- إذا كان هناك حقل أو أكثر من حقل تكون حقول اختيارية(optional fields) مثل : حقل رقم الفاكس .
- أن يكون لدي حقول متعددة القيم (repeating field) أي تحمل أكثر من قيمة مثل : رقم الهاتف .

كيف نعرف أطوال السجلات على اختلاف أنواعها ؟

بالنسبة للFixed Length Record يكون التعامل معها بسهولة فالأطوال ثابتة ولمعرفة الأطوال السجلات بخوارزميات .

ولكن في الVariable Length Record نجهل معرفة أطوال السجلات وبالتالي فإن الكمبيوتر يعرف ذلك بطرق منها :

- يوجد char مثل : ($ ، # ، ★) توضع عند نهاية الحقل للدلالة على نهايته .
- يحسب عدد الbytes المراد تخزينه ووضعه قبل السجل مباشرة بعد ذلك يُخزن في السجل فمثلاً



عدد جميع الbytes في المثال ب13، فيعرف من خلاله طول السجل قبل البيانات .

كتل السجلات والسجلات الممتدة والغير ممتدة :

Spanned Versus Unsnapped Records

كما ذكرنا أنه عند نقل البيانات بين القرص والذاكرة الرئيسية فإن ذلك يتم عن طريق الbuffer والذي يستطيع نقل أكبر وحدة وهي الblock .
لابد من وجود تناسق بين حجم السجل وحجم الblock ، فحجم الblock أكبر من حجم السجل وذلك لكي يتمكن من نقل السجل كاملاً إلا في حالات نادرة يكون فيها حجم السجل اكبر من حجم الblock .
عندما ينقل ال block السجل فإنه يكون هناك مساحات فارغة في الblock ولابد من استغلالها ولحساب هذه المساحة الفارغة عن طريق المعادلة الآتية:

r= record , b = block ,bfr = blocking factor for the file

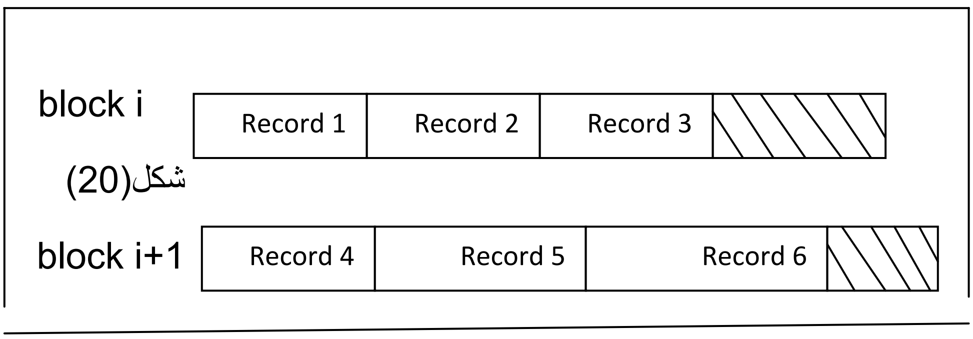
فإذا كان : b >= r

فإن : bfr = b / r

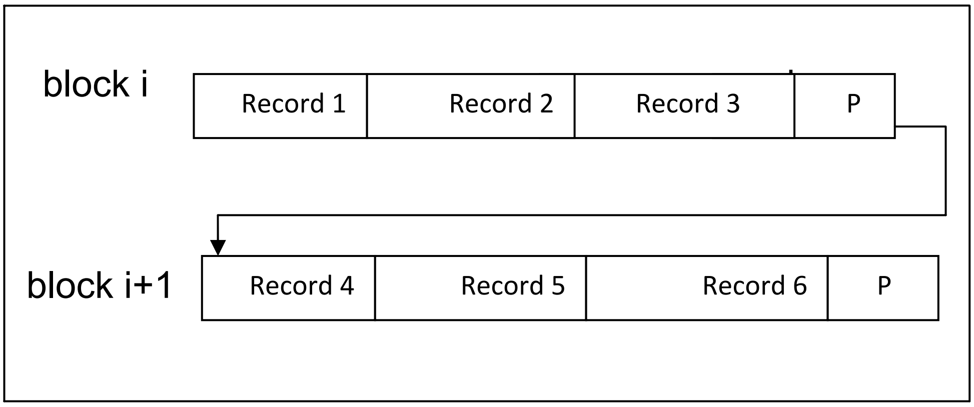
∴ المساحة الفارغة = (b – bfr) bytes .

ولكي نستفيد من المساحة الفارغة يمكننا تقسيم السجل وتخزين جزء من السجل في هذه المساحة المتبقية من الblock وتخزين القسم الباقي من السجل في block آخر مع وجود مؤشر في نهاية القسم الأول في الblock الأول ليؤشر للقسم الموجود في ال block الآخرالذي يحوي باقي السجل ، فكلما صغر السجل كلما أستفدنا من المساحة الخالية ويُعرف هذا التنظيم بالspanned لأن في هذه الحالة يمتد على أكثر من block واحد ، هنالك أنظمة تشغيل تسمح بذلك بينما هناك أنظمة أخرى لاتسمح بذلك وبالتالي أي لblock يحمل سجلاً كاملاً .

في الUnspanned Organaztion لايُسمح بتجاوز حدود الblock لتخزين سجلات الملف.



شكل(20)



شكل(21)

في شكل (20) نوع التنظيم Unspanned ، بينما في الشكل (21) نوع التنظيم spanned .

البيانات المخزنة هذه تنظم في شكل ملفات من السجلات File of record ولهذا يجب تخزين هذه السجلات بداخل الملفات بطيقة تُسهل من التعامل معها ، وهناك العديد من الطرق لتنظيم الملفات منها الheap file والsequential file وhashed file وسنتحدث عن هذه التنظيمات .

متى ما كان السجل أكبر من الblock لابد من استخدام الspanned organization

لماذا ؟؟؟؟

- في ال record variable length يُمكن استخدام تنظيم الspanned والunspanned .

تخصيص كتل الملفات على القرص

Allocation File Block on Disk

كيف تخزن كتل الملفات على القرص ؟

1. **طريقة الcontiguous allocation** : توضع الكتل (blocks) في القرص متتالية ، مما يجعل عملية القراءة للملف سريعة وذلك بإستعمال الdoble buffering، لكن بالمقابل تكون عملية الإضافة صعبة .

2. طريقة الlinked allocation : أي كتلة تحتوي على مؤشر يؤشر للكتلة التالية وهذه الطريقة تجعل عملية الإضافة سهلة ولكن عملية القراءة بطيئة جداً لأنه عشوائي ، على عكس السابقة .

3. طريقة الclusters of consecutive disk block : وهي عبارة عن هجين من الطريقتين السابقتين ، حيث تجمع الكتل المتجاورة بوحدة أكبر تسمى الcluster هذه التجمعات تتصل ببعضها يطلق عليها bile segment .

4. طريقة الindexed allocation : يستعمل فهرس ليؤشر إلى مواضع الblock،ومن الممكن أن نجد طريقة تضم عدة أفكار من الطرق السابقة .

بادئة الملف File Header :

بادئة الملف أو واصفة الملف تحتوي على معلومات عن الملف والتي تحتاج لها البرامج للوصول إلى السجلات بالملف .

البادئة تحتوي على معلومة عن عناوين ال block المكونة للملف ، وكذلك على وصف للشكل العام للسجل(record format description) والتي تحتوي على طول السجل بالإضافة لترتيب الحقول في السجلات والتي من النوع (fixed length unspanned) ، وتحتوي كذلك على نوع الحقل ، الفاصلات(separators) ونوع السجل للسجلات التي من النوع (variable length record) .

عندما نبحث عن سجل ما بالقرص لابد من نقل block واحد أو أكثر إلى الذاكرة ثم تقوم البرامج بالبحث عن السجل في الذاكرة بإستعمال المعلومات الموجودة في بادئة الملف ، فإذا كان عنوان السجل المطلوب غير معروف يكون البحث بالlinear search لكل الblocks حيث ننقل كل السجلات إلى الذاكرة حتى يتم العثور على السجل المطلوب أو ينتهي الملف وهذه العملية تأخذ وقت طويل للغاية .

يمكن تقسيم العمليات على الملفات الى مجموعتين رئيسيتين هما:

1- عمليات الإسترجاع (Retrieval operations) .

2- عمليات التعديل (update operations) .

عمليات الإسترجاع لاتقوم بأي عمليات على الملف ولكن تقرأ فقط السجلات المحددة ، أما في عمليات التعديل فهي تجري تعديل على الملف إما بلإضاف أو الحذف أو التعديل في حقل السجلات .

في كلا الطريقتين نحتاج لإختيار سجل أو سجلات بناءً على شرط اختيار معين ، فمثلاً في سجل لموظف حقوله هي name , ssn, salary , jop , department ; فيمكن أن يكون شرط الإختيار بسيط (simple selection condition) بعلامة التساوي (Equality operation) و مثلاً :

(salary = 500) أو (ssn = 1234) .

ويمكن أن يكون الشرط بسيط أيضاً بعمليات المقارنة (Comparison operation) مثلاً :

(salary > 250) .

عموماً يعتمد البحث على شرط الإختيار البسيط فإن كان الشرط معقداً فإن برنامج الDBMS أو المبرمج يقوم بتجزئته إلى شروط بسيطة تستعمل للوصول على السجل ، فمثلاً إن كان لدينا الشرط المعقد الآتي :

(name = ' ali ') and (salary >250)

فإن البحث سيتم أولاً بإختبار الشرط الأول وهو الname والتحقق منه ، وبعد الوصول إلى السجلات التي تحقق هذا الشرط ، سيتم بعد ذلك اختبار الشرط الثاني وهو الsalary .

إن كان هناك عدة سجلات تحقق شرط البحث فإن أول سجل نصل إليه يكون هو ال current record ، ثم ننتقل منه إلى بقية السجلات .

العمليات على الملف للوصول الى السجلات والعمل عليها تختلف على حسب نظم ادارة قواعد البيانات

الDBMS أو اللغة المستعملة ولكن على العموم يمكن حصرها كالآتي :

Open: تجهز الملف لعملية القراءة او الكتابة ويتم حجز عدد ملائم من الbuffer على الأقل اثنين لوضع الblock بها تقوم بقراءة الfile header ووضع مؤشر الملف في بداية الملف.

Reset : تضع مؤشر الملف المفتوح في بداية الملف .

(Find or locate) : تبحث عن أول سجل يحقق شرط البحث وتنقل ال block الذي يحوي السجل الذي يحقق الشرط ، ويقوم مؤشر الملف بالإشارة إلى السجل في الbuffer ليصير الcurrent record .

Red or (Get) : تنقل الcurrent record من الbuffer إلى البرنامج الذي يطلبه ، أيضاً قد تُحرك مؤشر الملف إلى السجل التالي والذي قد يكون مطلوب بعد هذا السجل .

Find Next : تبحث عن السجل التالي الذي يحقق شرط البحث ، تنقل الblock الحاوي لهذا السجل إلى الbuffer (إذا لم يكن أصلاً موجوداً بها) ، يوضع السجل في الbuffer ويصبح ال current record .

Modify : تعدل قيمة حقل السجل الحالي current record ثم تعدل الملف على القرص .

Delete : تمسح السجل الحالي current ثم تعدل الملف على القرص to reflect the modification .

Insert : تضيف سجل جديد بالملف حيث تحدد أولاً الblock الذي يدخل به السجل ، ثم تنقل هذا الblock إلى الbuffer (إن لم يكن موجوداً) تكتب السجل على الbuffer ثم تكتب محتوى الbuffer على القرص .

Close : إكمال عملية الوصول والتعامل مع الملف بتحرير الbuffer .

كل العمليات السابقة عدا (close & open) تُسمى Record at a Time Operation

لأن أي عملية تنفذ على سجل واحد ، يمكن أن نجمع العمليات Read , find next في عملية واحدة تطبق على الملف وهي scan ، وهذه المجموعة من العمليات تُسمى Set at Time Operation .

Scan : اذا تم فتح الملف في هذه اللحظة فان الscan ، تعطي أول سجل ، وفي غير ذلك تعطى السجل التالي . إذا تم تحديد شرط مع هذه العملية فهي تعطى أول سجل يحقق الشرط ثم التالي . وهكذا . وتوجد أيضا أمثلة للعمليات من النوع Set at Time Operation .

Find all : تحدد كل السجلات بالملف والتي تحقق الشرط.

Find Ordered : تسترجع كل السجلات بالملف بترتيب محدد.

عند اختيار تنظيم لأحد الملفات في البيانات يجب أن تاخذ العوامل السبعة الهامة التالية في الحسبان :

1- استرجاع سريع للبيانات .

2- مخرجات مرتفعة من العمليات الجارية لمدخلات تشغيل البيانات والصياغة .

3- استخدام كفء لمكان التخزين .

4- حماية من الفشل وفقدان البيانات .

5- تقليل الحاجة إلى إعادة التنظيم .

6- إمكانية شمول النمو المستقبلي .

7- الأمن من الإستخدام غير المخول به .

وعادة تتعارض هذه الأهداف ويجب أن إختيار تنظيم الملف الذي يوفر توازناً معقولاً عبر الوعايير الموجودة في الموارد المتاحة .

Reorganize : تبدأ عملية التنظيم كما سنرى لحقا بعض تنظيمات الملف تحتاج إعادة ترتيب دورياً ، فمثلاً ... إعادة ترتيب الملف بناءً على حقل آخر.

فيما يلي نميز بين مصطلحين وهما : Access Method & File Organization .

File Organization : تنظيم الملف تعني تنظيم البيانات داخل الملف إلى سجلات وblocks ونعني به أيضا هياكل الوصول إلى البيانات access structures ونعني به أيضاً طريقة وضع السجلات والblocks في وسائط التخزين .

Access method : طريقة الوصول إلى البيانات تضم مجموعة من العمليات التي تتيح التعامل مع الملف (مثل العمليات الموضحة سابقا)على كل حال يمكن أن نطبق عدد من Access method على تنظيم معين للملفات فمثل الindexed access لا يمكن أن يطبق إلا على الملف المفهرس أصلاً .

عادة نختار تنظيم الملف وطريقة الوصول حسب طبيعة العمل في الملف فمثل بعض الملفات تكونStatic بمعنى أن التعديل فيها ليذكر بينما بعض الملفات Dynamic بمعنى أن التعديل فيها مستمر،بعض الملفات نحتاج فيها لقراءة كمية من السجلات في اللحظة الواحدة .

مثلاً إذا حدد المستخدم أنه يبحث عن بيانات الموظف برقمه إذاً يقوم المبرمج ببناء الملف مفهرساً برقم الموظف أو نحتاج إلى الوصول للبيانات بالقسم فبالتالي نرتبها بالقسم وهذا ما سيتم توضيحه في الجزء الخاص بالفهرسة .

- انظمة الملفات File Organisation :-

ونقصد بها هياكل البيانات في التخزين الخارجي ، حيثُ تُنظم البيانات المخزنة على شكل ملفات من السجلات ، ولذلك لابد من تخزينها بطرق تُسّهل التعامل وإجراء العمليات على السجلات المخزنة ، وسنتعرف فيما يلي على طرق تنظيم الملفات ووضع السجلات فيزيائياً في الملفات وعلى تنظيمها بداخل القرص الصلب دون غيره من وسائط التخزين الخارجي للميزات التي ذكرناها سابقاً بالقرص الصلب.

- File Access :-

وهي طريقة الوصول إلى السجلات في الملف .

- عمليات الملفات File Opreation :-

هناك عمليات متعددة يمكن أن نجريها على الملفات منها :

- عملية الإسترداد (Retrieval Operation) : القراءة من الملف سواء قراءة للملف كاملاً أو البحث عن سجل معين ، ولاتقوم بأي تعديل على الملف أي أنها.
- التحديث (Updating Operations) من حذف ، وإضافة ، تعديل .
- المسح .

أهداف التنظيم للملفات :

هناك ثلاثة أهداف أساسية لأنظمة الملفات :

1. سرعة الوصول للمعلومات داخل الملف(البحث السريع عن السجلات) .
2. و الإستخدام الأمثل للتخزين الخارجي .
3. سهولة التحديث.

أنظمة الملفات كثيرة وسنتطرق لبعض من الأنظمة المنتشرة وذكر ميزاتها وعيوبها فهذه التنظيمات قد تُحقق هدف ما بكفاءة ولكن لهدف آخر ضعيفة ، وفيما يلي بعض أنظمة الملفات .

1- الملف النصي Text File :

الملف النصي هو ملف سهل التطبيق ولكنه لايحوي أي نوع من التصنيف للمعلومات ولذلك يتم التعامل معه كوحدة واحده عادة (يقرأ كله ويكتب كله) ويستخدم الملف النصي في بعض قواعد البيانات لحفظ حقول المذكرات.

2- الملف العمومي Pile File :

هو ملف يحتوي على سجلات لموجودات ذات بيانات مختلفة (لاتوجد أسماء ثابتة للحقول ولاأطوال ولاأنواع) وعليه عند الكتابة فيه نكتب مع كل حقل إسم الحقل .

مثال :

الإسم :: محمد علي	نوع السيارة :: تايوتا
العمر :: 24	الموديل :: 2003
الكلية :: الحاسوب	رقم السيارة :: 2435
الفرقة:: الرابعة	

الرمز ↑ = نهاية الحقل .

الرمز $ =نهاية الحقل .

الرمز ! = نهاية الملف .

وتكون إعادة صياغة الملف كالآتي :

الأسم↑ محمد علي↑ العمر↑24↑ الكلية↑الحاسوب↑ الفرقة↑ الرابعة↑ نوع السيارة↑ تايوتا↑

الموديل↑2003↑ رقم السيارة ↑2435↑$!

خوارزمية للكتابة في الملف العمومي Pile File -:

١. أنشيء ملف نصي وأفتحه .
٢. إقرأ إسم الحقل وأكتبه في الملف .
٣. إقرأ محتوى الحقل وأكتبه في الملف
٤. اكتب رمز نهاية إسم الحقل .
٥. أكتب رمز نهاية محتوى الحقل .
٦. إذا لم يكن هذا الحقل هو الحقل الأخير في السجل عد إلى ٢ .
٧. أكتب رمز نهاية السجل .

اكتب رمزا نهاية الملف .

ميزات وعيوب الملف العمومي Pile File :

يعتبر الملف مرن جداًبالنسبة لأطوال الحقول وأنواعها وعددها ، الاضافة فيه سهلة جداً وهذه من الميزات ، ويعيب هذا النوع من الملفات التعميم الشديد والذي يجعل الإسترجاع والتعديل بطيء جداً .

رغم أن فكرة الملف العمومي تبدو نظرية ولاتصلح لبناء قواعد البيانات إلا أنها مستخدمة في بعض بنوك المعلومات .

يمكن إستخدام الspecial charcter مثل ١ ، $ ، . . . . الخ لتدل على نهاية الحقل أو السجل أو الملف.

3. الملف الغير مرتب :

Serial Files (Unordered File)

يعتبر هذا ابسط نوع من أنواع تنظيم الملفات File Organization حيث تدخل السجلات الى الملف حسب ترتيب إدخالها فيه ، فالسجل الجديد يدخل في نهاية الملف ،وهو ملف يحوي سجلات من نوع واحد و محدد (ثابت) أي نقوم بتعريف السجل المراد التعامل معه ، كلما أردنا أن نضيف سجل نعلم سلفاً أن تعريفها محدد يسمى هذا النوع من الملفات heap file or pile file وهذا النوع يستخدم لإستعمالات خاصة مثل secondary indexes أو حفظ البيانات لإستعمال مستقبلي في هذا النوع من الملفات .

عمليات الإضافة و البحث والحذف في الملف الغير مرتب :

الإضافة : وجود الترتيب هو الذى يصعب الوصول الى الموقع المناسب ولذلك فان الاضافة الى سجل غير مرتب يكون اسهل فالسجل يتم تخزينه Block ونحن نسعى أن تكون عملية الإضافة سريعة في جميع أنواع التنظيم وخطوات الإضافة تكون كالآتي :

١- ينقل الblock الأخير من الملف إلى الbuffer .
٢- يضاف السجل الجديد إلى هذا الblock .
٣- يُكتب الblock مرة أخرى في القرص (Rewritten) .

*عنوان الblock الأخير من الملف يٌحفظ في الfile header .

البحث : كلما كانت عملية البحث سريعة كلما كان عملية الUpdate تكون أفضل وسريعة وهكذا وهي هنا عملية صعبة ومكلفة وذلك لأن البيانات غير مرتبة وللبحث لايمكن إستخدام الbainary Search لأن شرطها للبيانات المرتبة ونستخدم الlinear Search ويتطلب ذلك المرور على كل الblocks واحدة تلو الأخرى فإن كان هناك سجل واحد يحقق شرط البحث فيجب على البرنامج أن يقرأ نصف الblocks قبل أن يجد السجل المطلوب وفي حالة أن الشرط يتحقق في أكثر من سجل أو لا يوجد سجل يحقق الشرط فلابد من قراءة ال blocks بكامله للبحث عن السجلات.

عملية المسح :

1. نستخدم أولاً الLinear Search للبحث و لتحديد الBlock الذي يحدد ذلك السجل.
2. أنقل الBlock الذي يحوي السجل المطلوب إلى الbuffer .
3. أمسح السجل من الbuffer .

4. أعد كتابة ال block في الdisk .

وهذه آلية للdelete تترك فراغات في الblock .

وهنالك آلية أخرى وهي أنه أي سجل يُترك فيه bit او byte ونضع فيه علامة المسح (deletin mark) إما أن تكون (0) أو (1) تحسباً أنه عندما نجد القيمة (0) في علامة المسح فهذا يعني أنه محذوف أما إذا وجدنا القيمة (1) فهذا يعني أنه مخزن عليها ، الآلية الأولى والثانية تترك مساحات وللإستغلال الأمثل للمساحة الفارغة في الblock بعد الحذف تحتاج لإعادة تنظيم للملفات دورياً (reorganization) .

إعادة التنظيم للملفات الغير مرتبة :

هنالك آليات عدة لإعادة التنظيم منها :

- آلية تتم بقراءة الblocks على التوالي ،ثم تجمع السجلات الموجودة في الblocks من جديد ونلغي السجلات المحذوفة بعد اكتمال عملية إعادة التنظيم تكون المساحة بأفضل استغلال .
- آلية أخرى تستعمل مكان السجلات المحذوفة للكتابة فيها مرة أخرى ولكن هذه العملية تحتاج لمعرفة الأماكن الفارغة التي يمكن أن نكتب بها بكل الblocks.
- ويستعمل لتنظيم الملف الغير مرتب أيضاً الSpanned or Unspanned ويمكن استعمال الFixed or Variable Length Records وفيها أن عملية التعديل قد تتطلب مسح السجل القديم ثم إدخال السجل الجديد بنهاية الملف لأن السجل الجديد قد يكون أكبر من القديم وبالتالي لاتسعه المساحة القديمة .

لقراءة جميع السجلات بترتيب معين ننشئ نسخة جديدة من الملف مرتبة وعملية الترتيب للملف مكلفة كلما كان حجم الملف اكبر.

الملفات غير المرتبة إذا استعملت (fixed length records using unsnapped blocks and contiguous allocation) يكون الوصول الى أي سجل بموضعه في الملف فإذا كانت السجلات مرقمة 0 , 1 , 2 , ...,r-1. والسجلات داخل الblock ، أيضا مرقمة

0 , 1 , 2, bfr-1 (حيث bfr هو ال blocking factor) السجل رقم i في الملف يحدد موضعه كالتي:

- في block رقم (I / bfr) .
- وداخل الblock في سجل رقم (I mod bfr) .

هذا النوع من الملفات ولأن السجلات يمكن الوصول إليها حسب موضعها يسمى relative or direct .

مميزات الملف الغير مرتب:

١. سهولة الاستخدام لهذا الملف لوجود تعريف داخل السجلات .

٢. عدم تكرار كتابة أسماء الحقول(كما في الملف العمومي)
٣. الإضافة في الملف غير المرتب سهلة جداً(نضع السجل الجديد في آخر الملف(Appending) .

عيوب الملف الغير مرتب:

١. عادة ما تكون أحجامها محددة وهذه تعتبرمن المساوئ وذلك لأنه لايمكن تجاوز هذه الأحجام .
٢. عملية البحث ليست سريعة فلايمكن إستخدام طرق البحث السريعة مثل البحث الثنائي Binary Scearch وذلك لأن الملف غير مرتب .

➢ عند مقارنة الserial file مع الpile file نلاحظ أن تحديد أطوال الحقول يمكن من استخدام خوارزميات مبنية على عمليات حسابية بسيطة للوصول للسجلات المختلفة حسب ترتيبها ، كما يمكن حساب عدد السجلات في المقطع الواحد ومعرفة المساحات المطلوبة في التخزين الخارجي لحفظ الملف .

4- الملف المرتب (التتابعي) Sequential File :

هو ملف غير مرتب ولكن تم ترتيبه فيزيائياً بإستخدام أحد الحقول المكونة لهذا السجل يسمى حقل التعديل (Ordering Field) وللقيام بعملية ترتيب السجلات أي بإستخدام أحد الحقول كمفتاح ، وعندها نقول أن هذا السجل مرتب بإستخدام الحقل كذا وكذا وإذا فقد السجل هذا الحقل يكون السجل غير مرتب ، يكون الترتيب بإستخدام حقل واحد فقط وإذا كان عدد من الحقول يكون البحث عن طريقة الفهرس كما سيأتي لاحقاً في الملف المرتب المفهرس.

ميزاته:

١. السرعة في الاسترجاع وذلك لإمكانية إستخدام خوارزميات البحث السريعه مثل البحث الثنائي BinaryScearch ، ولكن تعتمد على حقل الترتيب .
٢. كل مساوئ الملف الغير مرتب تغير إلى محاسن .

عيوبه:

١. لايمكن أن أضيف إلا بعد تحديد الموقع الصحيح وهذه العملية تكون معقدة كلما كبر حجم السجلات لأننا نحتاج إلى قراءة نصف هذه السجلات .
٢. كما أن التعديل في حقل المفتاح قد يؤدي إلى إفساد الترتيب السجل .

تعتبر الملفات المرتبة أفضل من الملفات الغير مرتبة :

1- قراءة السجلات على حسب مفتاح الترتيب تكون سريعة جدا لأننا لانحتاج لإجراء عملية ترتيب للبيانات .

2- الوصول الى السجل التالي من السجل الحالي على حسب ترتيب المفتاح، غالباً لا يحتاج لقراءة block آخر من ملف وذلك لأن السجل في نفس ال block الحالي (إلا أن يكون السجل الحالي هو الأخير في الblock) بحيث لايوجد بعده سجل مخزن .

3- استعمال مفتاح الترتيب (ordering key) كشرط للبحث (Search condition) يؤدي لسرعة عملية البحث ، وذلك عند استخدامنا لخوارزميه البحث الثاني (Binary search)وهي أفضل من خوارزمية ال(linear search)، والتي تقوم بالبحث داخل السجلات واحداً تلو الأخرى .

عملية الإضافة والحذف في الملف المرتب مكلفة جداً لأن السجلات يجب أن تبقى Physically Ordere .

الإضافة :

لإضافة سجل يجب أولاً أن نوجد له المكان المناسب الصحيح في الملف بناءً على قيمة الحقل المفتاحي ، ثم نخلق مساحة لإضافة السجل وهذه العملية تكون أصعب كلما زاد حجم الملف لأنه في المتوسط نحتاج لإزاحة نصف السجلات لإيجاد المكان الفارغ للسجل الجديد وهذا يعني أن نصف الblocks تقرأ من الdisk ثم تُعاد كتابتها أي تُحرك لأسفل .

لزيادة الكفاءة في عملية الإضافة :

يمكن الإبقاء على بعض المساحة غير مستغلة في كل block تحسباً لإضافة سجلات لاحقاً ولكن هذا حل مؤقت فبمجرد إمتلاء هذه المساحة الفارغة تظهر المشكلة مرة أخرى .

وكذلك يمكن إنشاء ملف مؤقت غير مرتب لنكتب فيه مافاض عن كل block ولذا فهو ملف الفيضان وسنتحدث عن هذه الطريقة لاحقاً .

الحذف : يُمكن تلافي عملية الحذف وذلك بإستخدام الdeleteing marker ودورياً نقوم بعمل الreorganazation إعادة التنظيم كما استخدمته في الملفات الغير مرتبة .

عملية البحث في الملف المرتب :

تعتبر عملية البحث سريعة وذلك لإستعمال مفتاح الترتيب كشرط للبحث ، ونستخدم البحث الثنائي Bainary Search للملفات وتُجرى هذه العملية على الblocks ومنها نبحث وصولاً للRecord .

ليكن هنالك ملف مرتب به عدد من الblocks ومرتبة حسب مفتاح الترتيب ، وللوصول إلى البيانات المطلوبه يكون البحث عن طريق خوارزمية البحث الثنائية التالية .

خوارزمية البحث الثنائية :

السجل الأول = **F** السجل الأخير = **L**

السجل الأوسط = **midd** السجل المراد البحث عنه = **i**

```
While  (Last >=First ) do
Begin
midd := (First+Llast) div 2;
If (i < midd)  then
Last:= midd-1
else
If ( i > midd) then
First:=midd+1
else
If
(i=midd)then
go to found
else
go to not found;
end;
go to not found.
```

وفيما يلي برنامج فرعياً لعملية البحث الثنائي (Bainary Search) :

```
function binarysearch(var a:data;target:integer):
integer;
var first,last,midd:integer ;
begin
```

```
first:=1;
last:=n ;
repeat
midd:=(first+last) div 2 ;
if a[midd].no>target then
last:=midd-1
else
first:=midd+1 ;
until (first>last) or (a[midd].no=target ) ;
if a[midd].no=target then
binarysearch:=midd
else
begin
binarysearch:=0 ;
writeln('this target is not found ;
end ;
end ;
```

البحث يكون أكثر كفاءة عندما يكون متضمناً شروط بناءً على حقل الترتيب فلا فائدة من كون أن الملف مرتب بحقل(رقم الطالب) بينما البحث يكون بحقل آخر(الإسم) غير الحقل المفتاحي وإلا فإنه لابد لنا من عمل ملف آخر حقله المفتاحي يكون بالإسم أو نبحث في الملف القديم بطريقة الLinear Search .

تقوم خوارزمية البحث الثنائي بقراءة عدد (log2 b) من الblock في حالة وجود سجل أم لا ، بينما في خوارزمية الLinear search تقرأ عددblock (b/2) في المتوسط إذا وجد السجل وعدد b من الblock إذا لم يجد السجل .

مثال :

ملف عدد ال block به تساوي 120 block ، أحسب عدد الblock access إذا تم البحث عنه دخل سجل بداخل ملف وذلك في حالة :

1- الملف مرتب .

2- الملف غير مرتب .

الحل :

في الملف المرتب نستخدم خوارزمية البحث الثنائي للبحث عن السجل :

Block access= log2 b = log2(120) ➔ 7 blok access .

في الملف الغير مرتب فإنه في أفضل الأحوال وفي حال وجود سجل فإن

Block access = b/2 =120/2→60 block access

في حال المرور على كل السجلات وعدم وجود السجل المطلوب فإن

Block access = b → 120 block access

التعديل في الملف المرتب :

للتعديل في حقل معين يعتمد على عاملان أساسيان :

أ- شرط البحث الذي تصل به إلى السجل ويكون هذا الحقل يُعتمد عليه في عملية البحث فإن كان شرط البحث يعتمد على حقل المفتاح يمكن الوصول إلى السجل بإستعمال ال binary search ، فيما عدا ذلك يجب أن نستعمل الlinear search .

ب- الحقل المراد تعديله إن كان بخلاف حقل المفتاح فهو يُعدل ثم تعاد كتابته في نفس الموقع الفيزيائي(physical location) بإعتبار أننا نعمل على Fixed Length Record ، أما تعديل حقل المفتاح نفسه يعني أن السجل يجب أن يتعدل موضعه الحقيقي في القرص مما يتطلب مسح السجل القديم وإضافة السجل الجديد .

عملية القراءة :

بتجاهل ملف الفيضان تكون عملية القراءة سريعة جدا وذلك لن الblocks تقرأ بالتتالي باستعمال الdouble buffering أما بأخذ ملف الفيضان في الحسبان يجب أن :

1- يرتب ملف الفيضانOverflow أولاً .

2- دمج الملف الرئيسي مع ملف الفيضان للحصول على ملف جديد مرتب مع مراعاة حذف السجلات المؤشرة وهي (السجلات المحذوفة سابقاً وبها علامة Deletion marker)

لقراءة جميع السجلات بترتيب معين ننشئ نسخة جديدة من الملف مرتبة وعملية الترتيب للملف مكلفة كلما كان الحجم أكبر وهناك طرق(تقنيات) خاصة لترتيب الملفات ....الترتيب الخارجي(External Sorting) .

الترتيب الخارجي External Sorting :

وتكون للملفات المرتبة والمحفوظة في التخزين الخارجي والتي قد تحتاج لأضعاف مساحة التخزين الداخلي(الذاكرة الرئيسية) وبالتالي سنحتاج لإستخدام خوارزميات أخرى تسمى خوارزميات الترتيب الخارجيExternal Sorting Algorithms .

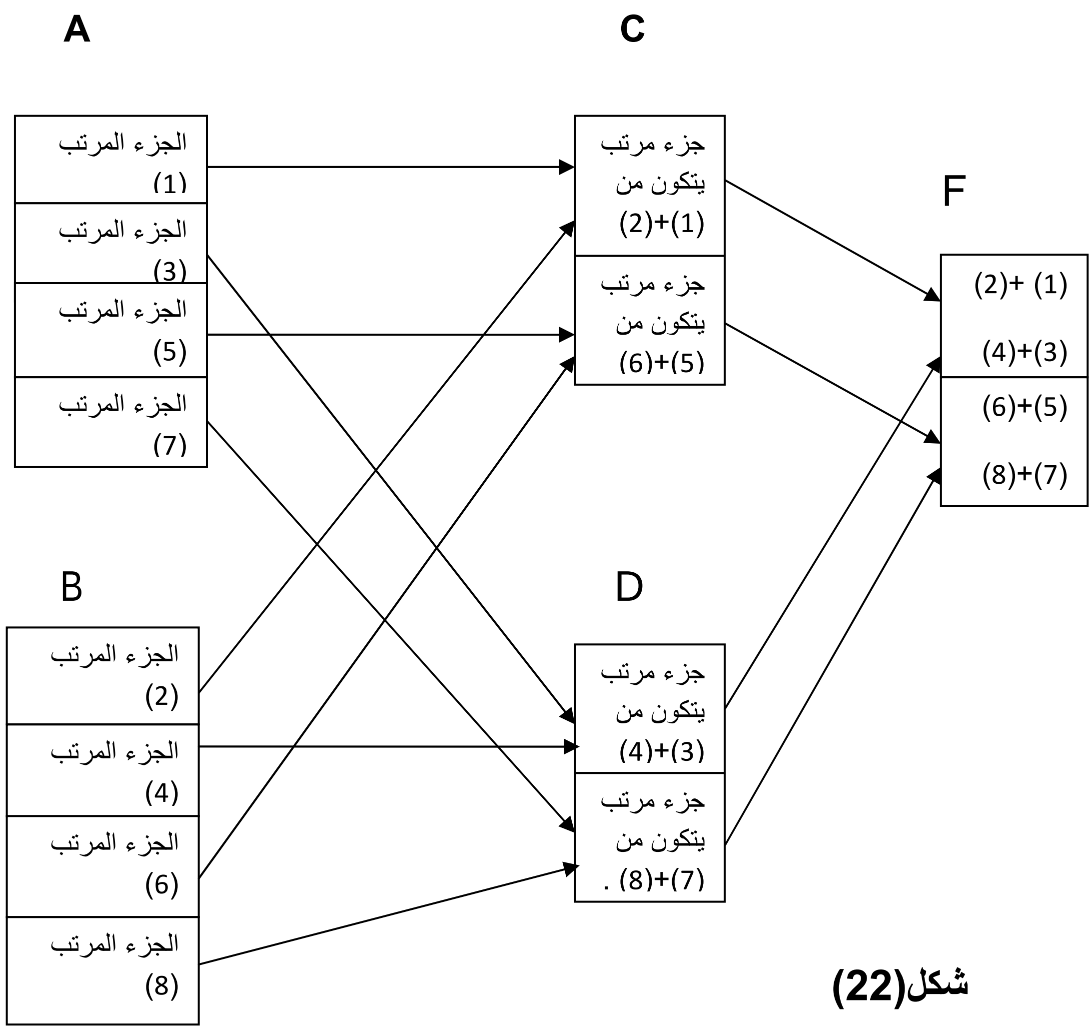
من خوارزميات الترتيب الخارجي :-

الدمج الثنائي (Tow Way Merge):-

هذا النوع من خوارزميات الترتيب الخارجي يتكون من جزئين رئيسين هما :

- الجزء الأول عملية تكوين الأجزاء المرتبة وهي عبارة عن ملء الجزء المتاح من الذاكرة الرئيسية بالسجلات وترتيب هذه السجلات ثم كتابتها في ملفين جديدين بالتناوب .

- الجزء الثاني عملية دمج الأجزاء المرتبة (Merging Process) من العملية السابقة خرجنا بملفين يتكون كل واحد منهما من سجلات مرتبة ، والآن نقوم بدمج هذه الأجزاء المرتبة في ملفين جديدين وتتكرر هذه العملية حتى نخرج بملف مرتب واحد.



شكل(22)

خوارزمية الدمج الثنائي :

هذه الخوارزمية تدمج الأجزاء المرتبة في الملفين A , B في ملف مرتب واحد وهو (out)

Two way merge(A , B ,out) :

```
Exchange :=true

Number_of_sorted partitions:=2

While (Number_of_sorted partitions > 1)
```

كرر حتى يصبح عدد الأجزاء المرتبة 1

```
If Exchange
```

ادمج الملفين A , B في C , D

```
Number_of_sorted partitions:=one_pass(a,b,c,d)

Exchange:=false

Else
```

ادمج الملفين C , D في A , B

```
Number_of_sorted partitions:=one_pass(c,d,a,b)

Exchange:=true

End if;

End while;

If exchange
```

حدد أين يكون الملف النهائي المرتب

```
Out :=A

Else

Out :=C

End two_way_merge
```

Function one pass(in1,in2,out1,out2)

تدمج هذه الأجزاء المرتبة من الملفين n1,n2 في الملفين out1,out2

```
Exchange:=true

Number_of_sorted partitions=0

While not eof(in1)and not eof(in2)

Number_of_sorted partitions = Number_of_sorted
partitions+1

If exchange

Merge(in1,in2,out1)

Exchange := false

Else

Merge(in1,in2,out2)

Exchange := true

End if;

End while

Return Number_of_sorted partitions
```

end one pass

والبرنامج الفرعي التالي لعملية الدمج الثنائي حيث يتم فيها دمج ملفين f1,f2 في ملف ثالث f3:

```
repeat
if st1.idno<st2.idno then
begin
write(f3,st1);
if not eof(f1) then
read(f1,st1);
end
else
```

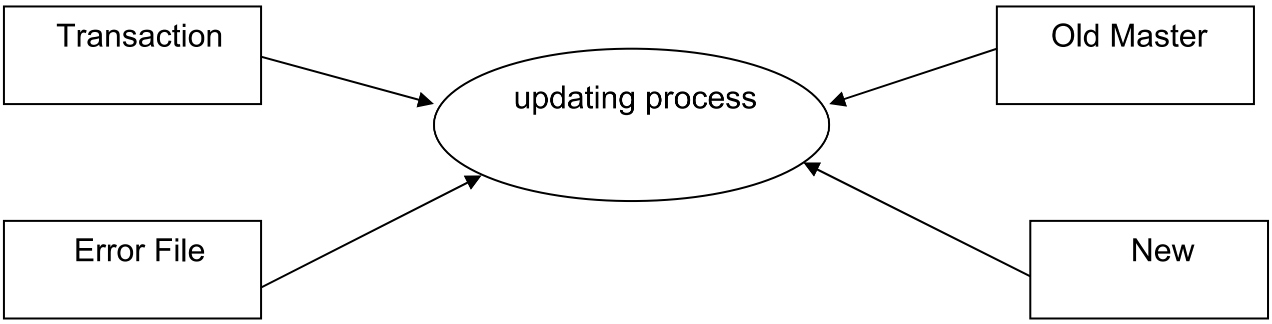
```
if st1.idno=st2.idno then
begin
write(f3,st)
if (not eof(f1)) and (not eof(f2)) then
begin
read(f1,st1 );
read(f2,st2);
end;
end
else
begin
write(f3,st2);
if not eof(f2) then
read(f2,st2);
end;
until (eof(f1)) and (eof(f2));
if st1.idno<st2.idno then
begin
write(f3,st1);
write(f3,st2);
end
else
begin
write(f3,st2);
write(f3,st1);
end;
```

تحديث الملف المرتب (Sequntial File Updating) :

ذكرنا أن مساوئ الملف المرتب أن التعديل في حقل المفتاح قد يفسد الترتيب ، وكذلك إضافة سجل جديد إلى نهاية الملف المرتب لذلك كان لابد من حل لهذه المشكلة وذلك بإستخدام طرق للتحديث لاتُفسد الترتيب ومنها التحديث الدوري ، طريقة الفراغات ، طريقة الفيضانات .

أ- التحديث الدوري (Batch Updating Process):-

وهي الطريقة الأولى من طرق التحديث وفيه يتم تجميع التعديلات كلها في ملف جديد يسمى ملف الحركة transaction file ويسمى الملف الرئيسي المرتب master file ، وتنفذ هذه العملية دورياً(يومياً ، اسبوعياً ....الخ) حسبما يحتاج النظام وهذه الطريقة تصلح للتطبيقات التي لاتحتاج لعكس التعديلات فوراً في الملف الرئيسي.



شكل(23)

ملف الحركة Transaction File :

يحوي نفس الحقول الموجودة في الملف الرئيسي مع إضافة حقل يسمى حقل التعديل ،ويكون نوع حقل التعديل عادة char حيث يكتب فيه أحد هذه الرموز :

I : تعني الإضافة insert .

D : تعني المسح delet .

C : تعني التعديل change .

ويكون ملف الحركة مرتب بنفس المفتاح الذي رتب به الملف الرئيسي.

خوارزمية التحديث الدوري(Batch Updating) :

اقرأ سجل من الملف الرئيسي القديم في M ثم اقرأ سجل من ملف الحركة في T ، قارن المفتاح في T مع المفتاح في M .

(أ) إذا كان T.key < M.key أوجدي الآتي :

(i) إذا كانت قيمة حقل التعديل i اكتب السجل T في الملف الرئيسي الجديد وإلا فإن هنالك خطأ فأكتبه في ملف الأخطاء .

(ii) اقرأ سجل جديد في T من ملف الحركة ثم عد إلى 2 .

(ب) إذا كانت M.key = T.key أوجدي الآتي :

(i) إذا كانت قيمة حقل التعديل D اقفز إلى (iv) .

(ii) إذا كانت قيمة حقل التعديل C اكتب السجل T في الملف الرئيسي الجديد ثم اقفز إلى (iv) .

(iii) إذا كانت قيمة حقل التعديل ليست C ولا D فإن هنالك خطأ واكتب في ملف الأخطاء السجل T ، وأما السجل M فأكتبه في الملف الرئيسي الجديد .

(iv) اقرأ سجل جديد من ملف الحركة في T واقرأ سجل جديد من الملف الرئيسي القديم في M ثم عد إلى 2 .

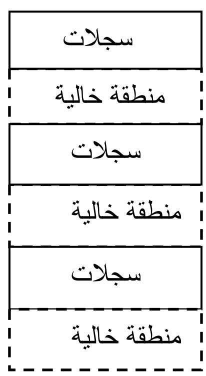
(ج) إذا كان T.key > M.key :

اكتب السجل M في الملف الرئيسي الجديد واقرأ سجل جديد من الملف الرئيسي القديم في M ثم عد إلى 2 .

كرر حتى نهاية الملف .

ب- طريقة الفراغات :

وهي طريقة أخرى للتحديث الدوري حيث يتم فيها تقسيم الملف إلى مقاطع وكل مقطع يحوي عدد من السجلات (N سجل) ونقوم بترك عدد محدود من السجلات في المقطع الفارغ (مثلاً إذا كان عدد السجلات 7 في المقطع الواحد نقوم بترك سجلين فارغين) ، بهذه الطريقة يمكننا الإستفادة من هذه الفراغات لإجراء عمليات الإضافة في مواقعها الصحيحة .



شكل(24)

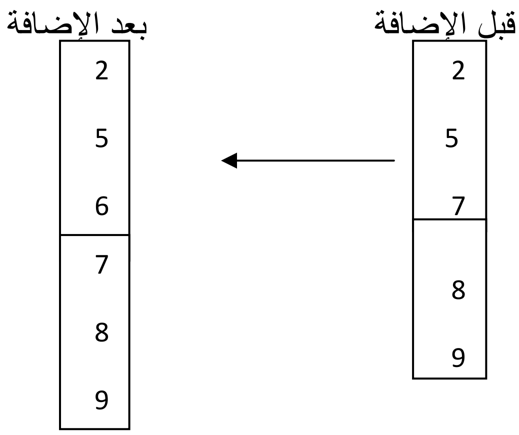
ويعيب هذه الطريقة أن كثافة البيانات في الملف ضعيفة .

وتتم الإضافة في الملف الرئيسي بخلاف الطريقة القادمة.

ليكن لدينا مقطع محدد وأردنا الإضافة له يكون هناك خياران:

١- إما بتحريك السجل الأخير إلى المقطع التالي .

مثلاً إذا أردنا إضافة المفتاح رقم 6 :

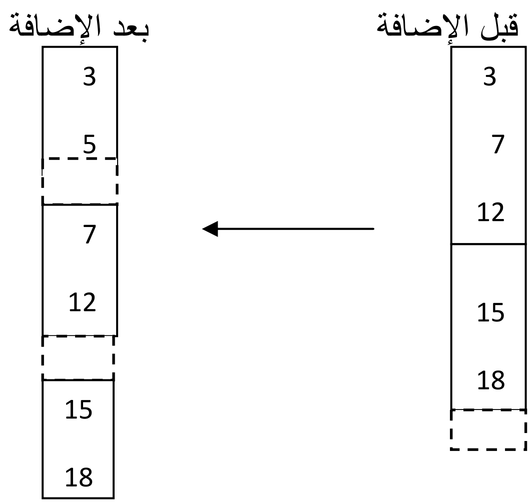


شكل (25)

هنا تم وضع المفتاح 6 في موقعه الصحيح بعد نقل المفتاح 7 إلى المقطع اللاحق .

٢- **وإما** بإنشاء مقطع جديد خالي ثم تُحول نصف سجلات المقطع الممتلئ إلى المقطع الجديد.

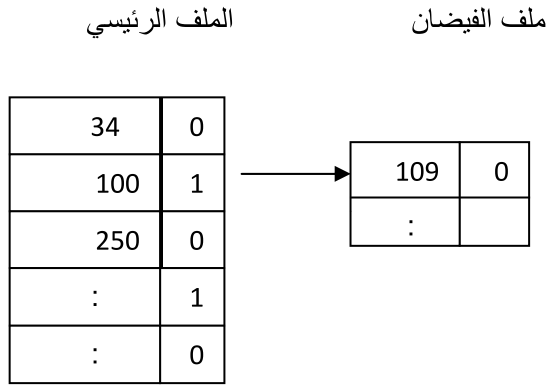
عند إضافة المفتاح رقم 5 يتم التحديث بإنشاء (حجز) مقطع جديد ونقل المفتاح 7 و 12 إلى المقطع الجديد كما في الشكل التالي:



شكل (26)

ج- طريقة الفيضان :

تٌنشئ ملف آخر يسمى ملف الفيضان Over flow File وتٌضيف حقل جديد لسجلات الملف يسمى مؤشر الفيضان(over flew pointer)، وبالتالي فإن الملف الرئيسي يكون ثابتاً في هذه الطريقة وعدد السجلات لايزيد وتكون الزيادة في ملف الفيضان ، بخلاف طريقة الفراغات فتتم كل الإضافات في الملف الرئيسي .

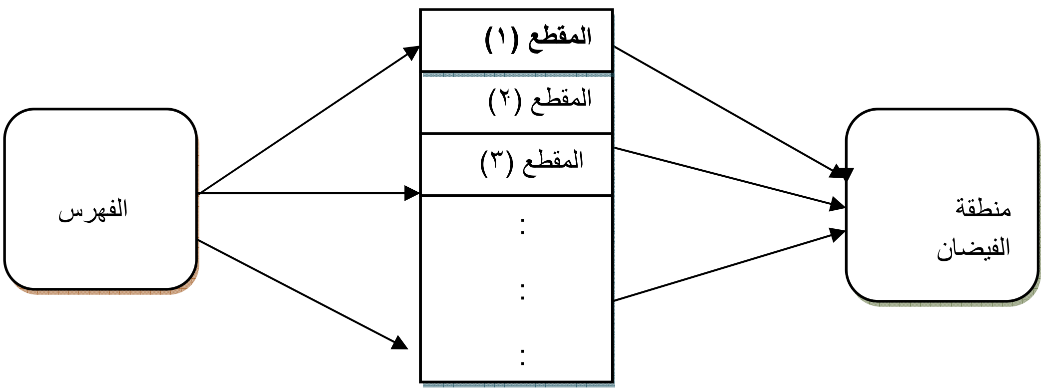


في هذا المثال أضيف السجل 109 إلى منطقة الفيضان ووضع مؤشراً له في السجل 100.

٤- الملف المرتب المفهرس (Indexed Sequential File)

يتم الوصول لهذا النوع بإضافة مكونين أساسيين للملف التتابعي وهما:

- الفهرس (Index) وتتم إضافته لزيادة سرعة البحث.
- منطقة الفيضان (Overflow Area) وذلك لمعالجة عمليات التحديث.



مفهوم الفهرس :

الفهرس هو عبارة جدول (ملف) يتكون كل سجل من سجلاته من حقلين :

- حقل يحمل قيمة المفتاح والذي نستخدمه في عملية البحث .
- حقل عبارة عن مؤشر لرقم المقطع الموجود في أوله السجل الذي يحمل المفتاح .

سجل الفهرس يُسمى بالمدخل (entry) .

كيف يتم إنشاء الفهرس :

- من كل مقطع في الملف الرئيسي نأخذ رقمه ومفتاح السجل الأول فيه ونكتبه في الفهرس ، وبهذا يكون كل مدخل يحوي مفتاح ورقم مقطع وتكون المداخل في الفهرس مرتبة بنفس ترتيب الملف الرئيسي .
- الجدول (الملف) الذي تم إنشاؤه في الخطوة السابقة يُسمى بالمستوى الأول في الفهرس ، وبما أنه ملف فهو أيضاً يتكون من مقاطع وعليه في هذه الخطوة ننشأ فهرس للمستوى الأول بنفس الطريقة الموصوفة في الخطوة السابقة ونسميه المستوى الثاني للفهرس .

ويستمر ذلك حتى نتحصل على مستوى يتكون من مقطع واحد .

مثال :

لدينا ملف مرتب به 000 1000 سجل وطول السجل الواحد فيه 200 byte وطول المفتاح فيه 14 byte ، فإذا استخدمنا نظام طول المقطع فيه 2000 byte ونحتاج لـ6 byte لكتابة رقم المقطع فأحسب :

a) عدد مستويات الفهرس وطوله الكلي .
b) عدد عمليات القراءة إذا بحثنا في هذا الملف بإستخدام البحث الثنائي .

الحل :

⇦ الطول الكلي للملف = عدد السجلات * طول السجل الواحد .
⇦ 00 1000 * 200 = 000 000 200 byte .

عدد المقاطع = الطول الكلي للملف / طول المقطع .

⇦ 000 000 200 / 2000 = 000 100 مقطع .

عدد السجلات في المقطع = طول المقطع / طول السجل

⇦ 2000 / 200 = 10 سجلات .

عدد المقاطع في الملف الرئيسي = عدد السجلات / عدد السجلات في المقطع

⇦ 1000 000 / 10 = 100 000 مقطع .

عدد المداخل في المستوى الأول من الفهرس .

طول المدخل = طول المفتاح + طول المؤشر .

⇦ 14 + 6 = 20 byte .

عدد المداخل في المقطع بالنسبة للفهرس (BFI)= طول المقطع / طول المدخل

⇦ 2000 / 20 = 100 مدخل .

إن عدد المقاطع في المستوى n يُساوي عدد المداخل في المستوى n+1 وعلى هذا نجد أن :

عدد المقاطع في المستوى الأول = عدد المقاطع في الملف الرئيسي / BFI

⇦ 100/100 = 1000مقطع .

∴ عدد المداخل في المستوى الثاني من الفهرس =1000 مدخل .

عدد المقاطع في المستوى الثاني= عدد المقاطع في المستوى الأول / BFI

⇦ 1000 /100= 10مقطع .

∴ عدد المداخل في المستوى الثالث من الفهرس =10 مدخل

عدد المقاطع في المستوى الثالث= عدد المقاطع في المستوى الثاني/ BFI

⇦ 10 /10= 1 مقطع.

∴ عدد المداخل في المستوى الرابع من الفهرس= مدخل واحد .

هذا الفهرس يتكون من ثلاثث مستويات .

b)عدد عمليات القراءة بإستخدام البحث الثنائي $\lceil \log_2 10^N \rceil$

إذن عدد عمليات القراءة $\lceil \log_2 10^6 \rceil$ = 20 .

هذا الرمز $\lceil \ \rceil$ يعني أن نكمل الرمز واحد صحيح مثلاً $\lceil 5.12 \rceil$ = 6 .

وهذا الرمز $\lfloor \ \rfloor$ يعني أن نتجاهل الكسر مثلاً $\lfloor 5.87 \rfloor$ = 5 .

بينما عدد عمليات القراءة بإستخدام الفهرس $\lceil \log_i M \rceil$ +1

حيث :

N = عدد السجلات في الملف .

M = عدد المقاطع في الملف .

i = مُعامل التجميع للفهرس (طول المقطع / طول المدخل) .

منطقة الفيضان (overflow area) :

كما أسلفنا لأن الملف المفهرس المرتب يحوي ترتيب فلابد من طريقة إضافة لاتفسد الترتيب وأنسب طريقة من الطرق التي شرحناها مع الملف التتابعي لهذا الملف هي طريقة الفيضان.

إعادة التنظيم (reorganization) :

إذا تضخم جدول الفيضان فإننا نبني الملف من جديد في خطوتين :

a. بدمج الملف الرئيسي مع منطقة الفيضان .
b. بناء فهرس جديد بنفس الطريقة القديمة التي تم بناء الفهرس القديم .

عمليات الملف الفهرس :

i. **الإضافة :** وتتم الإضافة بأن نبحث حتى نجد المقطع المناسب في الملف الرئيسي، فإن تم العثور على المكان المناسب فيتم حجزه ، وإن لم يوجد فإننا نقوم بوضع السجل المراد إضافته في منطقة الفيضان ويكتب رقمه في حقل الفيضان في السجل السابق له في الملف الرئيسي .
ii. **المسح :** إذا وجد السجل المراد حذفه فإنه يتم مسحهُ .
iii. **البحث :** البحث في منطقة الفيضان بطئ ولذا كان لابد من إعادة تنظيم للملف وذلك في حالة أن يحصل تضخم لجدول الفيضان وتكون خوارزمية البحث في الملف الرتب المفهرس كما يلي:

حيث : n = رقم المقطع الموجود في المستوى الأخير من الفهرس .

1) إقرأ المقطع n .

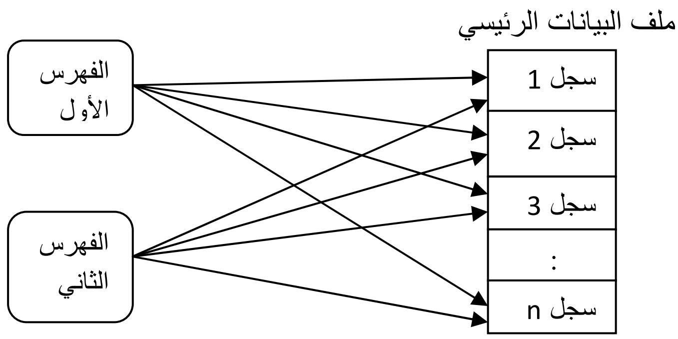
2) n = المؤشر الموجود في المدخل الذي يحوي أكبر مفتاح ، أصغرمن أو يُساوي المفتاح المطلوب.

3) كرر 1 و 2 حتى تصل لمؤشر لمقطع بيانات .

٤) إقرأ المقطع n وأبحث فيه عن السجل المطلوب .

5- الملف المفهرس (Indexed File) :

نحتاج إلى بناء فهرس لكل حقل من حقول البحث وذلك لتصميم ملف يمكن البحث فيه بأكثر من حقل ،مثلاً البحث عن بيانات طالب عن طريق اسمه ورقم بطاقته وعنوانه. هذا النظام يختلف عن السابق بكونه لايوجد ترتيب للملف ، كما أن المؤشر في مدخل الفهرس لايؤشر إلى مقطع بل إلى سجل .



لابد من تحديث الفهرس ليعكس عمليات المسح والإضافة ،ولكي يكون الفهرس متحركاً نضطر إلى تقليل كثافته وذلك بترك مساحات فارغة ولتتم فيها الإضافة ونترك هذه المساحات الفارغة في جميع مستويات الفهرس ، وبصفة عامة إذا امتلأ مقطع في المستوى n نوجد مقطع جديد في نفس المستوى n ونكتب مدخلاً لهذا المقطع في المستوى n+1 يؤشر إلى المقطع الجديد وهذا يُساعد كثيراً في عملية الإضافة .

مقارنة بين الملف المرتب المفهرس والملف المفهرس:

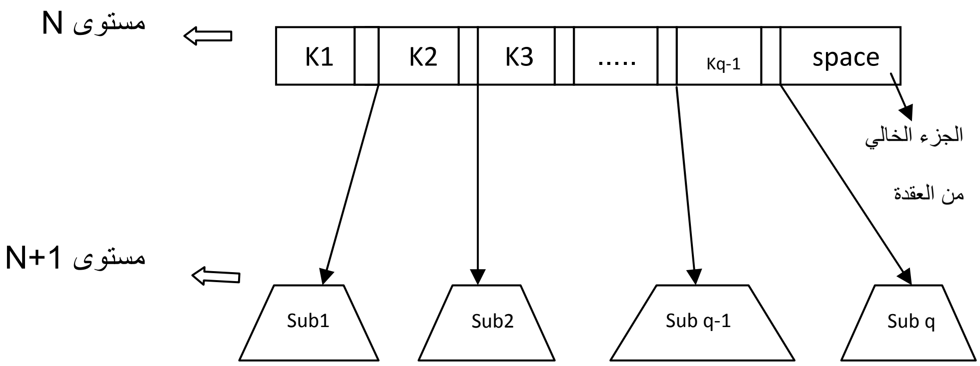
الملف المرتب المفهرس	الملف المفهرس
يوجد به حقل واحد للبحث .	يوجد به عدة حقول للبحث .
لابد أن يكون الملف الرئيسي مرتباً (Sequential).	الملف الرئيسي غير مرتب(serial) لوجود عدة مفاتيح .
المؤشر في مدخل الفهرس يؤشر إلى مقطع .	المؤشر في مدخل الفهرس يؤشر إلى سجل .
لابد من وجود منطقة فيضان لكي تتم الإضافة من دون إفساد الترتيب .	لاتوجد منطقة فيضان بل تتم الإضافة مباشرة إلى الملف الرئيسي .
يعتبر فهرس ساكناً أي لايتم عكس عمليات التحديث بل يتم التغيير عند عملية إعادة التنظيم .	يعتبرفهرساً متحركاً تنعكس عليه عمليات التحديث .

الشجرة البائية (B_Tree) :

الفهرس في الملف المفهرس يكون عبارة عن شجرة بائية (B tree) ، وهي عبارة عن شجرة من الدرجة t نطبق عليها شروط محدودة الغرض منها أن تكون شبه متوازنة ، فهي شجرة لها عقدة جذر ، وتتميز هذه الشجرة بالصفات التالية :

- بالنسبة للعقدة تكون المفاتيح فيها مرتبة ترتيباً تصاعدياً ، وتسمى العقدة بورقة leaf إذا كان ليس لديها أبناء (أي أن درجتها تساوي الصفر)، وإلا فإنها تسمى بعقدة داخلية internal ماعدا عقدة الجذر root فهي عقدة مصممة بشكل خاص .
- المؤشرات بالنسبة للعقدة الداخلية x تُشير بعدد n(x)+1 إلى كل ابن من ابنائها ، بينما في العقدة الورقية فإن مؤشراتها تُشير إلى nil فهي ليس لها أبناء .
- كل الأوراق (نهايات الشجرة) لها نفس الإرتفاع أي بمستوى واحد، والذي هو نفسه ارتفاع الشجرة .
- لتخزين عدد من المفاتيح في العقدة فإن المتغير t يتحكم بالحد الأدنى والأقصى ، حيث أن ال2 > t ، وأصغر درجة للشجرة البائية هي 2 .
- الحد الأدنى للمفاتيح t – 1 لكل العقد ماعدا عقدة الجذر فهي تحتوي على مفتاح واحد كحد أدنى .
- الحد الأقصى للمفاتيح 2t – 1 لكل العقد ، وتوصف العقدة بأنها ممتلئة إذا كان عدد المفاتيح بها 2t – 1 .

الشكل العام للعقدة في الB_tree :



K_i = قيم مفاتيح البحث حيث أنها تكون على الترتيب $k_1 < k_2 < k_3 < \ldots < k_q$.

Sub_i = الشجيرات الفرعية .

البحث في الأشجار b_tree :

للبحث عن المفتاح x فإنه يُقارن x ابتداءً من الجذر حتى يصل إلى الهدف ويرجع لنا المؤشر يؤشر إلى العقدة التي تحتوي على المفتاح في حال وجوده ، وإن لم توجد فإنه يؤشر إلى nil حيث أنه يتم اتخاذ القرار في تفرع متعدد ويعتمد التفرع على المفاتيح الموجودة للعقدة المعينة .

خوارزمية البحث:

إبدأ من الجذر ، إستخدم مفتاح البحث للوصول إلى الورق، إذا كان مفتاح البحث مساوي للجذر إذا تم إيجاد السجل وإلا إذا كان أقل من الجذر إذهب المؤشر اليسار وإلا إذا كان أكبر من الجذر إذهب إلى مؤشر اليمين.

```
Search: Start at root; use key comparisons to go to
leaf.
If search key is equal to root node then record is
found else
If its less than root node go to the left pointer
else if greater than go to the right pointer
```

وفيما يلي برنامجاً فرعياً لعملية البحث :

```
procedure searchnode(target:integer;p:pointer;var
found:Boolean ; var k:position );

begin
with p^ do
  if target<key[1] then
  begin
    found:=false؛
    k:=0؛
    end

  else  begin
          k:=count؛
          while (target<key[k]) and (k>1) do
            k:=k-1؛
           found:=true؛
          target:=key[k]؛
writeln('yes this key is found, target')؛

end
end؛
```

```
procedure search(target:integer;root:pointer;var
found:Boolean;var targetnode:pointer; var
targetpos:position );
begin
if root=nil then

  begin
  found:=false؛
  writeln('no key in this tree);
  end

 else

   begin
        searchnode(target,root,found,targetpos);
        if found then
          targetnode:=root

        else
          tnode,targetpos)
      end
end؛
```

والشكل التالي يوضح عملية البحث في الأشجار المتوازية :

فإذا كان لدينا الشجرة المتوازية الآتية :

وإذا أردنا البحث عن العدد ٢٢ نجد أن :

ثانيا: عملية الإضافة: insert operation

خوارزمية الإضافة:

١- أوجد الموقع الصحيح للمفتاح (بإستخدام عملية الإضافة).
٢- أضف المفتاح المراد إضافتة إلى العقدة الصحيحة.
٣- إذا كان هناك مساحة كافة ، تمت العملية وإلا إذهب إلى الخطوة التالية.
٤- إقسم العقدة إلى عقدتين
أعد توزيع المداخل بإنتظام بين العقدة الحالية والعقدة الجديدة.
٥- أضف العنصر الوسط إلى عقدة الأب وأضف العنصرين الذين يلي العنصر الوسط إلى العقدة الجديدة.
٦- إذهب إلى الخطوة رقم ٣.

1) Find correct leaf node

2) Add index entry to the node

3) If enough space, *done*!

4) Else, *split* the node

Redistribute entries evenly between the current

node and the new node

5) Insert <*middle key, ptr to new node*> to the parent

6) Go to Step 3

وفيما يلي برنامجاً فرعياً لعملية الإضافة :

```
Procedure
pushin(x:integer;xr,p:pointer;k:position);
var
i:position؛
begin
  with p^ do
    begin
      for i:=count  downto  k+1  do
        begin
          key[i+1]:=key[i؛[
          branch[i+1]:=branch[i];
        end؛

      key[k+1]:=x؛
      branch[k+1]:=xr؛
      count:=count+1
    end
end؛

procedure
split(x:integer;xr,p:pointer;k:position;var
y:integer ؛var yr:pointer);
var
I,mediam:position؛
begin
if k<=min then
  mediam:=min
else
  mediam:=min+1؛
new(yr);
with p^ do
  begin
    for i:=mediam+1 to max do
    begin
      yr^.key[i-mediam]:=key[i] ؛
      yr^.branch[i-mediam]:=branch[i]
    end؛
```

```
    yr^.count:=max-mediam؛
    count:=mediam؛
    if k<=min then
      pushin(x,xr,p,k(
    else
      pushin(x,xr,yr,k-mediam؛(
    y:=key[count؛[
    yr^.branch[0]:=branch[count؛[
    count:=count-1
  end
end؛

procedure pushdown(newkey:integer;p:pointer;var
pushup:79ddress ؛var x:integer; var xr:pointer);
var
k:position؛
found:79ddress؛
begin
if p=nil then
  begin
    pushup:=true؛
    x:=newkey؛
    xr:=nil
  end

else
  begin
        searchnode(newkey,p,found,k);
        if found the
       begin
          writeln('Error:inserting dublicate
key);
             write('enter new key to
insert:',newkey2 );

pushdown(newkey2,p^.branch[k],pushup,x,xr)
          end
        else
        if pushup then
        with p^ do
          if count<max then
          begin
            pushup:=false؛
            pushin(x,xr,p,k(
```

```
          end

        else  begin
                pushup:=true؛
                split(x,xr,p,k,x,xr)
              end
      end
end؛


procedure insert(newkey:integer;var
root:pointer);
var
pushup:80ddress؛
 x:integer؛
 xr,p:pointer؛
begin
pushdown(newkey,root,pushup,x,xr);
if pushup then
  begin
    new(p) ؛
    with p^ do
      begin
        count:=1؛
        key[1]:=x؛
        branch[0]:=root؛
        branch[1]:=xr؛
        root:=p
      end
  end
end؛
```

والشكل التالي يوضح عملية الإضافة :

ومثال آخر :

مثال رقم ٣:

ثالثًا: عملية المسح :Delete operation

خوارزمية عملية المسح:

١- أوجد العقدة الورقية الصحية (بإستخدام خوارزمية البحث).

٢- إمسح السجل المراد إزالتة.

٣- إذا كانت العقدة التي تم داخلها عملية المسح نصف إمتلاء و تمت عملية المسح ، إذا كانت غير ذالك إذهب إلى الخطوة التالية:

٤- إذاكان هناك إمكانية الاستعارة سجل من العقدة الشقيق ،تم إذا لم يمكن ذلك إذهب إلى الخطوة التالية:

٥- إدمج العقدة الحالية مع الشقيق

٦- إحذف الفاصل بين العقدة الحالية والعقدة الشقيق و العقدة الأصل.

٧- إذهب إلى الخطوة رقم ٣.

1) Find correct leaf node

2) Remove the entry from the node

3) If the node is at least half full, *done*!

4) Else, possibly *borrow* some entries from a sibling

5) If not possible, *merge* the node with the sibling

6) Delete the separator between the node and the

sibling from the parent node

7) Go to Step 3

وفيما يلي برنامجاً فرعياً لعملية الحذف :

```
 procedure remove(p:pointer;k:position);
var
i:position؛
begin
with p^ do
  begin
    for i:=k+1 to count do
    begin
```

```
      key[i-1]:=key[i] ؛
      branch[i-1]:=branch[i]
    end؛
    count:=count-1
  end
end ؛


procedure successor(p:pointer;k:position)؛
var
q:pointer؛
begin
q:=p^.branch[k]؛
while q^.branch[0]<>nil do
  q:=q^.branch[0]؛
p^.key[k]:=q^.key[1]
end ؛


procedure moveright(p:pointer;k:position)؛
var
c:position؛
begin
with p^.branch[k]^ do
  begin
    for c:=count downto 1 do
    begin
      key[c+1]:=key[c] ؛
      branch[c+1]:=branch[c[
    end؛
    branch[1]:=branch[0]؛
    count:=count+1؛
    key[1]:=p^.key[k]
  end؛

with p^.branch[k-1]^ do
  begin
    p^.key[k]:=key[count]؛
    p^.branch[k]^.branch[0]:=branch[count] ؛
    count:=count-1
  end
end؛

procedure moveleft(p:pointer;k:position)؛
```

```
var
c:position؛
begin
with p^.branch[k-1]^do
  begin
    count:=count+1؛
    key[count]:=p^.key[k]؛
    branch[count]:=p^.branch[k]^.branch[0]
  end؛

with p^.branch[k]^ do
  begin
    p^.key[k]:=key[1]؛
    branch[0]:=branch[1]؛
    count:=count-1؛
    for c:=1 to count do
    begin
      key[c]:=key[c+1]؛
      branch[c]:=branch[c+1]
    end
  end
end ؛

procedure combine(p:pointer;k:position )؛
var
c:position؛
q:pointer؛
begin
q:=p^.branch[k ]؛
with p^.branch[k-1] do
  begin
    count:=count-1؛
    key[count]:=p^.key[k] ؛
    branch[count]:=q^.branch[0]؛
    for c:=1 to q^.count do
    begin
      count:=count+1؛
      key[count]:=q^.branch[c]
    end
  end؛

with q^ do
  begin
    for c:=k to count-1 do
```

Wejdan Al-Haj

```
    begin
      key[c]:=key[c+1];
      branch[c]:=branch[c+1]
    end;
    count:=count-1
  end;
dispose(q)
end;

 procedure restore(p:pointer;k:position );
begin
if k=0 then
  if p^.branch[1]^.count>min then
    moveleft(p,1)
  else
    combine(p,1)
  else
  if k=p^.count then
          if p^.branch[k-1]^.count>min then
            moveright(p,k)
          else
            combine(p,k)
  else  if p^.branch[k-1]^.count>min then
          moveright(p,k)
  else  if p^.branch[k+1]^.count>min then
          moveleft(p,k+1)
  else
        combine(p,k)
end ;


procedure recdelete(target:integer;p:pointer;var
found:Boolean );
var
k:position;
begin
if p=nil then
  found:=false
else  with p^ do
      begin
        searchnode(target,p,found,k );
        if found then
          if branch[k-1]=nil then
            remove(p,k)
```

```
            else  begin
                    successor(p,k)؛

 recdelete(key[k],branch[k],found) ؛
                        if not found then
writeln('Target is not found in the 91ddress
node);
end
   else
   recdelete(target,branch[k],found)؛
           if branch[k]<>nil then
             if branch[k]^.count<min then
               restore(p,k)
       end
 end ؛

 procedure delete(target:integer;var root:
 pointe) ؛
 var
 found:91ddress؛
 p:pointer؛
 begin
 recdelete(target,root,found )؛
 if not found then
   writeln('Target is not found')

 else  if root^.count=0 then
       begin
         p:=root؛
         root:=root^.branch[0 ]؛
         dispose(p)
       end
 end ؛
```

والشكل التالي يوضح عملية الحذف :

مثال:

مثال ٢:

مثال٣:

مثال ٤:

الملف المباشر Direct File :

الملف المباشر يحتوي على نظام يحدد عنوان السجل المطلوب من المفتاح المباشر ويمكن أن نصل إلى ذلك بعدة طرق منها :

a) أن يكون المفتاح هو عنوان السجل :

في هذه الطريقة نُنشئ حقل تكون قيمته هي عنوان السجل ، مثلاً يمكننا أن نحفظ سجلات مجموعة من الطلاب في ملف ثم بعد ذلك نحدد لكل طالب رقم جلوس مطابقاً لرقم سجله .

b) بإستخدام علاقة أو قانون يحول المفتاح لعنوان :

تُستخدم إذا كان لدينا مفتاح متناسق

مثال :

افرض إننا نريد حفظ سجلات طلاب في جامعة بها سبعة كليات وكل كلية بها 5 سنوات ، وعدد الطلاب في السنة الدراسية الواحدة لايزيد عن 99 طالب ، فإذا صممنا الشكل التالي للمفتاح :

رقم الطالب	السنة الدراسية	رقم الكلية

يكون بإمكاننا كتابة خوارزمية تحول هذا المفتاح إلى رقم سجل محدد داخل الملف .

c) بإستخدام تقنية البعثرة (Hashing techniques) :

ترتكز فكرتها على الHashing والتي توفر سرعة في الوصول للسجلات ، حيث أنها تطبق دالة الhash على مفتاح السجل ويعطينا العنوان للسجل (أي أنها تحول الملف إلى عنوان) وتتم عملية التحويل في الذاكرة الرئيسية ولاتحتاج لأكثر من قراءة واحدة ، ينقسم نظام البعثرة إلى نظامين داخلي وخارجي .

نظام البعثرة الداخلي (Internal Hashing) :

هنا تطبق الدالة من خلال استخدام مصفوفة سجلات ، ولنفرض أن المصفوفة مكونة من M record بالترقيم 0,1,....,M-1 ، ولدينا M خانة بالعناوين المقابلة للترقيم ، نختار

الhash function لتحويل قيمة الhash field لقيمة صحيحة في المدى 0,….,M-1 ومن أشهر دوال hash function الدالة :

$$H(k) = k \bmod M$$

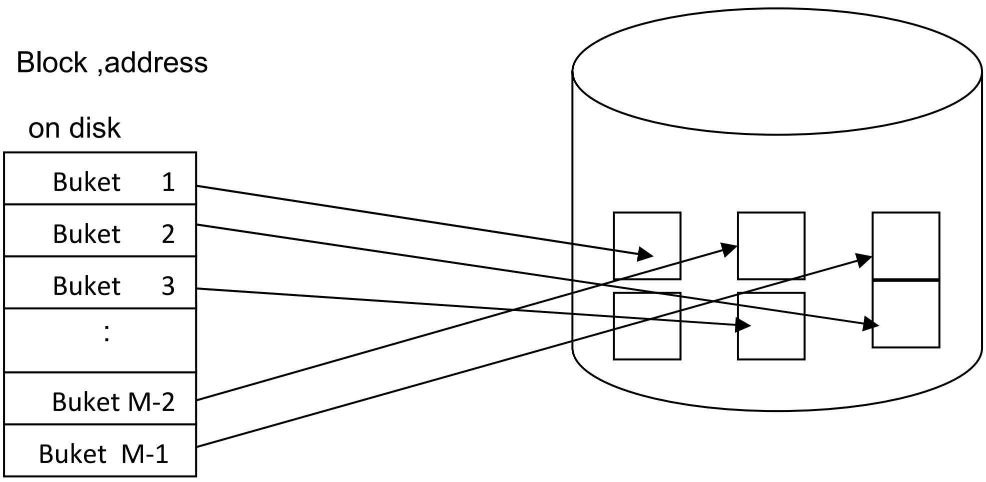
بالنسبة لقيم الحقول الغير رقمية كالحروف مثلاً يتم تحويلها لinteger قبل تطبيق الmod باستخدام الAscII code المقابل لها .

يُستخدم نظام البعثرة الداخلي للحفظ والقراءة في جداول موجودة داخل الذاكرة الرئيسية، مثلاً (symbol table in compiles) .

نظام البعثرة الخارجي (External Hashing) :

تتم عنونة المساحات المقسمة لBuckets وذلك لملائمة خصائص القرص ، فالBuckets عبارة عن block واحد أو مجموعة من الblocks المتجاورة والتي تضم مجموعة من الrecord .

دالة الhash function تنتج الkey وتقوم بربطه برقم الBucket الذي له علاقة برقم الkey الناتج بالإضافة لقيامها بتحديد رقم ال block داخل الBucket .



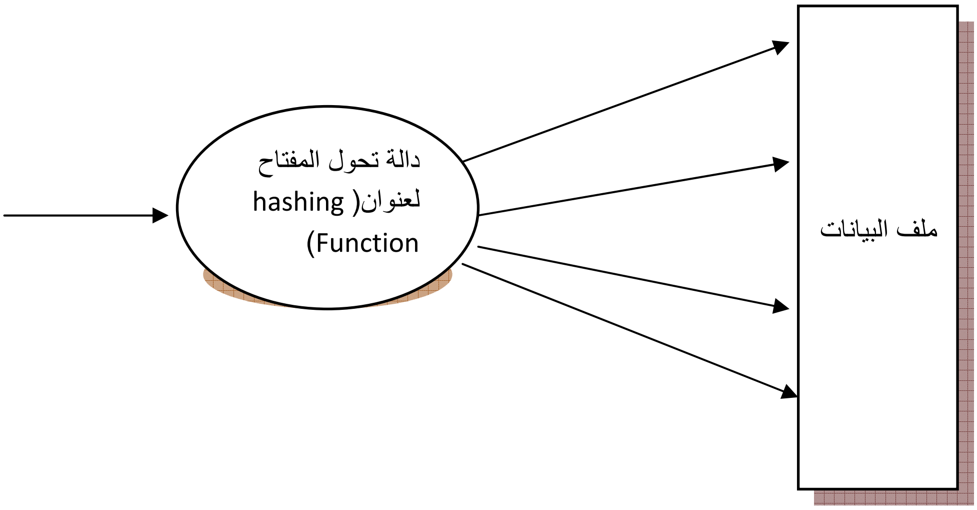
Matching buckets numbers to disk block addresses .

كيف تتم الكتابة في الملف المبعثر :

في البداية نحجز مساحة في التخزين الخارجي وتكون عادة أكبر من الطول المتوقع للملف الذي نريد حفظه ثم بعد ذلك إذا أردنا حفظ أي سجل في هذا الملف نقوم بتطبيق دالة البعثرة (Hashing function) على مفتاح السجل، والتي ترجع عنواناً داخل المساحة التي حجزناها في البداية ، ثم نحفظ السجل في هذا العنوان .

كيف تتم القراءة في الملف المبعثر :

نُطبق دالة البعثرة على المفتاح الذي نريد البحث فيه ، وترجع لنا الدالة نفس العنوان الذي سبق وأرجعته عند حفظ السجل ثم نقرأ السجل من هذا العنوان .



عيوب الملف المبعثر :

*المشكلة الرئيسية في هذا المفتاح هي التصادم Collision ونعني به أن تُرجع الدالة نفس العنوان لأكثر من مفتاح .

* المشكلة الثانية أن كثافة الملف عادة تكون قليلة ، ولذلك علاقة مباشرة بمشكلة التصادم ، وذلك لأن الملف كلما قلت كثافته قلّ احتمال حدوث التصادم .

حلول المشكلة :

1- العنوان المفتوح : وفي هذه الطريقة نضع السجل الجديد في العنوان اللاحق ، فمثلاً إذا كان العنوان الذي حدث فيه التصادم هو 101 نضع السجل في الموقع 102 وإذا تكرر التصادم نضع السجل الجديد في الموقع 103 وهكذا .

2- الفيضان ، وذكرنا هذه الطريقة من قبل .

3- استخدام أكثر من دالة بعثرة .

مثال :

صمم برنامجاً لبعثرة أسماء 30 طالب في مصفوفة تحوي 41 موقعاً واستخدم طريقة العنوان المفتوح (open address) لمعاجة مشكلة التصادم .

الحل :

نفرض أن الإسم 20 حرفاً .

```
▪   Function hash_adddress(name):integer
       Temp:=1
       For i:= 1 to 20 do
       Temp := temp * ord(name[i])
       Hash_address :=temp mod 41
End hash_address
▪        Write(name)
            I:=hash _address(name)
            While A[i]occupied do
            I:=(i+1)mod 41
            A[i]:=name
         End write
▪        Search(name)
            I:=hash_address(name)
            While A[i] & A[i]occupied
            I:=(i+1)mod 41
            If A[i] not occupied then
               Notefound
            Else
              Display name
         End search
```

مدا